Appropriate information system development

A methodology for sustainable cross-cultural information system production and use

Proefschrift

ter verkrijging van de graad van doctor

aan de Radboud Universiteit Nijmegen

op gezag van de rector magnificus prof. mr. S.C.J.J. Kortmann,

volgens besluit van het college van decanen

in het openbaar te verdedigen op dinsdag 25 september 2012

om 10.30 uur precies

door

Markus Pscheidt

geboren op 12 juli 1975

te Graz, Oostenrijk

Promotor:

Prof. dr. ir. Theodorus Petrus van der Weide

Copromotor:

Dr. Eduard Jozef Simons

Manuscriptcommissie:

Prof. dr. Ruerd Ruben

Dr. Nicolaas Moens (International Institute for Communication and Development)

Dr. Aldo de Moor (CommunitySense)

ISBN: 978-94-6191-436-1

Appropriate information system development

A methodology for sustainable cross-cultural information system production and use

Doctoral thesis

to obtain the degree of doctor

from Radboud University Nijmegen

on the authority of the Rector Magnificus prof. dr. S.C.J.J. Kortmann,

according to the decision of the Council of Deans

to be defended in public on Tuesday, September 25, 2012

at 10.30 hours

by

Markus Pscheidt

Born on July 12, 1975

in Graz, Austria

Supervisor:

Prof. dr. ir. Theodorus Petrus van der Weide

Co-supervisor:

Dr. Eduard Jozef Simons

Doctoral Thesis Committee:

Prof. dr. Ruerd Ruben

Dr. Nicolaas Moens (International Institute for Communication and Development)

Dr. Aldo de Moor (CommunitySense)

Appropriate information system development:
A methodology for sustainable cross-cultural information system production and use

Abstract

Appropriate information system development:
A methodology for sustainable cross-cultural information system production and use

by

Markus Pscheidt

IS projects have a low success rate. Even completed projects often have difficulty to be sustained. This is true in all parts of the world. However, success and sustainability are particularly problematic in the context of developing countries, where projects face increased complexity. Therefore IS projects in developing countries provide opportunities to learn from extreme cases.

In distributed software development, that is, in projects with actors from diverse locations and backgrounds, cultural, geographic and time distances become particularly evident. If some of the actors have limited experience with IS, then the challenge for collaborative IS development intensifies. Thereby, not only scarce resources, but also diverse understandings of success can put cross-cultural IS projects in jeopardy.

For less experienced IS users and developers it may seem beyond reach to actively engage in IS design. But to shape their own future, local participants not only need to acquire access to ICT; both users and developers need to become active producers of the technology they need. Otherwise they remain on the wrong side of the digital divide, and effective technology use remains indeed out of reach.

This thesis attempts to shed some light on how information system projects can actively integrate and empower local, less experienced participants into the development process, both at the user and the software developer levels. The result is a methodology that allows to tap external expertise as required and to establish local capacity over time.

The thesis draws on information systems, development studies and software development literature. An emphasis is put on Appropriate Technology principles and Open Source as a way to nurture international collaboration and settle issues of control and ownership of IS artifacts.

To my wife Nadalina,

for walking by my side.

To my parents,

for always being there.

Contents

List of Figures

List of Tables

Acronyms

AICT Appropriate ICT

AISD Appropriate Information System Development

ANT actor-network theory

AR Action Research

ARIS Academic Registry Information System

e-SURA electrónico - Sistema Universitário de Registo Acadêmico

CI Community Informatics

CPaL Community Participant Levels

DC Developing Countries

DD Distributed Development

DI Development Informatics

DSDM Dynamic Systems Development Method

DSR Design Science Research

ERP Enterprise Resource Planning

FDD Feature-Driven Development

FLOSS Free/Libre and Open Source Software

FOSS Free and Open Source Software

FSF Free Software Foundation

HDI Human Development Index

HISP Health Information System Project

ICT Information and Communication Technology

ICTD Information and Communication Technology and Development

ICT4D Information and Communication Technology for Development

IID Iterative and Incremental Development

IS Information System

ISDC Information Systems in Developing Countries

ISDT Information System Design Theory

ISPU Universidade Politécnica

MDGs Millennium Development Goals

MEC Ministry of Education and Culture

MIS Management Information Systems

OLPC One Laptop Per Child

OPUS Open University Systems

OSS Open Source Software

PaSS Participant Support System

SC Social Capital

SDLC Systems Development Life Cycle

SEComm Supporting and Enabling Communities framework

STS Socio-Technical Systems

SW Software

UCM Universidade Católica de Moçambique

UEM Universidade Eduardo Mondlane

UMBB Universidade Mussa Bin Bique

UN United Nations

UNDP United Nations Development Programme

UP Universidade Pedagógica

XP Extreme Programming

Account

Chapters 1 and 2

Markus Pscheidt and Theo van der Weide (2010b). "Sustainability of collaborative information system development projects – a North-South case study." In: *4th IDIA Conference: Exploring Success and Failure in Development Informatics: Innovation, Research and Practice*. Cape Town, South Africa

Chapter 3

Markus Pscheidt (2008). "Sustainability Factors for Information Systems in Developing Countries – Academic Registry Information System in Mozambique." In: *Prato Community CIRN Conference 2008: ICTs for Social Inclusion: What is the Reality?* Prato, Italy

Chapter 4

Markus Pscheidt, Victor van Reijswoud, and Theo van der Weide (2009). "Assessing Appropriate ICT with ARIS case in Mozambique." In: *ICCIR'09: 5th Annual International Conference on Computing and ICT Research*. Makerere University. Kampala, Uganda

Chapters 4 and 5

Markus Pscheidt and Theo van der Weide (2010a). "Bridging the Digital Divide by Open Source." In: *International Journal of Innovation in the Digital Economy, Special Issue on Digital Divide* 1.2, pp. 44–60

Chapter 5 and 6

Markus Pscheidt, Eduard J. Simons, and Theo van der Weide (2009). "Towards 'best practices' in North-South Open Source projects – lessons learned from the ARIS project in Mozambique." In: *3rd IDIA Conference: Digitally Empowering Communities: Learning from Development Informatics Practice*. Berg-en-dal, Kruger, South Africa

Chapter 7

Markus Pscheidt and Theo van der Weide (2009). "Supporting the ARIS community system in Mozambique." In: *Prato Community CIRN Conference 2009: Empowering communities: learning from community informatics practice*. Prato, Italy

Chapter 8

Markus Pscheidt (2011). "Structurational analysis of cross-cultural development of an academic registry information system in Mozambique." In: *Information Technology for Development* 17.3, pp. 168–186

Preface

Personally speaking, 2007 was a fruitful year. I could prepare the ideas and make the right contacts for writing this thesis, and I met my wife. Both have a lasting and still ongoing impact. Even though the thesis is now written, it has been a journey with a lot of learning and new insights involved, which hopefully did not only produce certain theoretical output, but also had some practical effects in the country that has accepted me to stay for seven years.

I'd expect that the experience of thesis writing will have an impact on future endeavors, be it through the particular subject-specific knowledge acquired, or be it thanks to certain capabilities that were inevitably practiced during the act of thesis writing; for example how to digest large amounts of information, how to put ideas into writing, and how to work with apparently conflicting theories, facts and behaviors. These are skills that should be useful in many life situations. For the improvements in such areas I'm as grateful as for the many insights in the field of study.

Concerning the specific subject matter, the thesis provided the opportunity to get a view of areas that are often less accessible to information technologists. For one, it was the vast area of development informatics – an emerging field that tries to integrate development studies with information and communication technology. Second, it was possible to learn about the more social aspects of informatics: Information systems are the about the effects that occur at the interplay of the development of technology and its organizational implementation. It was interesting to note that socially and technically oriented IS related research communities sometimes seem to lack a certain level of knowledge exchange. Similar insights are sometimes produced by both sides. For example, socio-organizational research is increasingly suggesting that IS development needs to be more adaptive to particular contexts, instead of aiming at universal solutions – similar to the agile software development

related research.

Working in the context of development countries, one has to constantly reflect one's own ways of thinking and ways of doing. This is a dynamic environment, and if one thing is for sure, it is constant change. Work, and life in general, has never become monotonous or let alone boring. It is a continuous challenge to keep up and at times get ahead in order to contribute something that has an impact to local development – not only as a short-term relief for the immediate moment, but something that facilitates structural changes with a developmental potential for longer periods in the future.

From the perspective of the distant observer in the 'rich' countries it is often difficult to assess if there is any positive development going on in the poorer countries of the world, and if the large sums of development funds that are being invested in fact have any sensible impact. After all, the media are full of troublesome news, especially from Africa. There is no simple answer to this question, and different regions struggle with different problems. My own experience in Southern Africa, particularly Mozambique, during my stay of seven years is that certain things are indeed moving forward, which is visible for example in improved infrastructure such as roads and fiber optics networks for Internet connectivity, the increasing exploitation of natural resources by local and international investors, by an emerging middle class. It can also be noted in increasing student numbers in public and private universities in major cities, but also in more rural areas, which were deprived of higher education until recently. But also in such mundane things as the banknotes used in everyday life there has been a big step forward within the time span of only a few years that saw the transition from notes of low quality, worn out, often glued together by adhesive tape, sometimes with a distinct smell, to state-of-the-art notes with modern security features.

Countries such as Mozambique, one of the poorest in the world, can demonstrate high annual growth rates, but naturally coming from a very low level, as indicated for example by the UN development index. And its development is accompanied by a constant struggle for good governance, to combat corruption, to empower civil society. It is here that foreign contributions through development cooperation, but also through foreign investment, can make a difference. The outcome of the current development processes may to a considerable degree be uncertain, but with the facilitation towards a democratic and self-determined society it will benefit not only the local population, but also those foreign individuals, organizations and countries who have or want to build economic and other kinds of relationships.

It is my hope that the research and the accompanying practical project described in this thesis provide a positive contribution to cooperation efforts between 'north' and 'south'. The research is concerned with ways to enable such cooperation in the realm of information system development. It builds upon the opportunities provided by open source to bring together strengths of participants in different parts of the network. It is oriented towards global equality, and in this respect it is satisfying to hear news such as the recent one from the International Monetary Fund that argues for greater income equality to achieve sustained economic development [1]. Although the studies referred to in this particular news item refer to equality within countries, I wouldn't be surprised if the world wouldn't benefit from more equality on a global level. But such a statement admittedly is already somewhat of a personal speculation, which shall take a backseat in the remainder of the thesis.

[1] http://www.imf.org/external/pubs/ft/fandd/2011/09/Berg.htm

Part I

Description of the research project

Chapter 1

Introduction

When introducing information systems in developing countries a lot can go wrong. To avoid getting lost in a myriad of interwoven problems, it is essential to be aware of pitfalls and to minimize risks. It is easy to lose user acceptance during the process, and once it is lost it can be hard to regain it. A particular challenge is to sustain working information systems over long periods of time. Abandoning half-finished or even fully developed systems implies wasted energy and resources, thus creating high opportunity costs.

At the outset of an information system initiative the user organization is challenged with understanding its requirements, procuring a proper technical solution and preparing the necessary conditions within the organization. These processes can be challenging. In many cases, organizations start over and over again in search of appropriate information systems that are capable to match their set of requirements. The introduction of a new information system often necessitates organizational development such as human resources capacity building. But which and how much competency is needed within the organization and what is realistic given the local conditions?

Given the scarcity of local resources in developing country contexts, information systems are often implemented with outside partners, frequently within the domain of development cooperation projects between developed and developing country partners. This constellation yields special opportunities, but also particular challenges.

In north-south projects there is often a difference not only in knowledge but in culture. Donors tend to be goal-oriented, whereas local people in developing country settings are rather process-oriented. This leads to a culture clash. Also, oral traditions play a large role; such traditions make it hard to demonstrate results (Anton Luger, personal

communication). Facing such cultural differences, achieving project goals and transferring knowledge between the cooperation partners can be challenging. Furthermore, at the end of the cooperation project, when external funds dry up, local stakeholders are often unprepared to take over the responsibilities to maintain a soundly functioning information system, not to mention its further functional and technical evolution.

Given this point of departure, the purpose of this thesis has been chosen to identify ways to improve sustainability – and thereby the overall success – of information system projects in contexts of scarce resources, where user organizations have yet limited experience with information systems development and use. The way that this challenge is approached is within collaborations of actors with varying degree of IS project experience, that is the capacity to develop and implement IS. In this thesis such a constellation is investigated within the context of a north-south development cooperation project. More specifically, the practical case that provides the real-world motivation and the opportunity for evaluation of research ideas and propositions is an international development cooperation project between universities in Mozambique and The Netherlands.

An immediate issue of concern is the resource scarcity, particularly with respect to the knowledge and skills of human resources. Any attempt for long-lasting IS use needs to be concerned with how to raise the level of human capacity. In the case of IS, there are two main groups: IS users and IS developers. Both need appropriate skills that are often not present in scarce resource contexts. The required skills include technical skills, but the two groups also need communication skills to exchange relevant information with each other.

If geographically distant organizations work together, then there is the implicit challenge to coordinate work between participants. This includes the practical issues of how to split functionality between collaborators, and how to resolve questions of ownership. Naturally, there is also a cultural distance between partners with different backgrounds. This can include language barriers. Cultural distance presents a particular challenge to produce relevant IS solutions.

Critical for the user acceptance and for the probability of long-term use is furthermore the provision of effective support. Different user organizations have different internal capacities. Some might be able to resolve a lot of the emerging difficulties concerning usage and technical maintenance internally, thus achieving autonomy more easily. Others have a stronger need for external support. In this respect, some questions emerge: Which level of support is best suited to solve emerging problems, for example at the level of internal

support, or by a regional support center? A fundamental question concerns financial viability of support structures and the attractiveness of the support services so that they are potentially sought by IS user organizations.

Therefore, long-term success in IS projects depends on a variety of factors, including capacity building, appropriate system design and effective support structures. This introductory chapter will provide a deeper look at problematic issues concerning success and sustainability. It will also give an overview of the research project, including context and goals. Before going into details, a general motivation and justification for ICT oriented research in developing countries is presented.

1.1 The value of development informatics research

Information systems, and more generally ICTs, have been put forward as important facilitators for development by many international organizations, and have become the focus of research published in several dedicated journals. Although the debate whether ICTs are indeed relevant to developing countries has been resolved "with a clear yes answer" (Walsham and Sahay, 2006, p. 7), this view is not beyond controversy among practitioners and researchers. Heeks (2010, p. 629) characterizes the history of ICTs in development as a cycle of "heavy over-promising followed by noticeable under-delivery"; the strong initial hype was followed by reports of a lot of hardly used or abandoned ICT projects, and only a minority of successful projects.

ICTs may indeed have been oversold, for example by touting that they would enable developing countries to *leapfrog* stages of economic development directly into the information society. To be more specific, technology leapfrogging indicates the implementation of a new technology without deploying the previous version of that technology, with the aim of accelerating development and promoting economic growth. An example of leapfrogging is the introduction of cellular telephone technology in a geographic area that never had a fixed line network established. In this way, at least one generation of technology is omitted. There are opportunities associated with technology leapfrogging. However, deploying the latest technology on its own does not automatically solve problems; it needs to be integrated with the social context and stakeholders. Otherwise there is a high probability of equipment being ineffectively used, misused or abandoned altogether (Davison et al., 2000).

One argument that has frequently been brought forward against ICTs in the de-

velopment arena is that people living in absolute or relative poverty better be assisted concerning their immediate needs, i. e. by providing food, clothing, housing, health care and education, rather than with costly information and communication technology. This line of criticism questions ICT investments in general or at least calls for trade-offs between specific ICT investments and alternative investments (Wade, 2002). However, effects of investing solely in first-order investments such as food, clothing, housing, health care and compulsory education are limited by an impact threshold, above which increasing investments cease to affect human development measures (Ngwenyama et al., 2006). The impact threshold can be overcome by complementary second-order investments. Second-order investments are investments that aim to provide opportunities for people to escape their hardships. They include investments in ICTs, post-secondary education, or economic literacy. Morawczynski and Ngwenyama (2007) have shown that although ICTs in isolation are not enough to impact human development, they have a strong positive influence if combined with investments in education and health care. Combining second-order with first-order investments can have significant impact on Human Development Index (HDI) measures. Given the impact of these second-order investments on human development, they recommend that "national policymakers should not undermine the importance of investments in areas such as ICT" (p. 8).

On the whole, development informatics research tends to accept that ICT potentially contributes to economic growth (Avgerou, 2008), which is a widely used, albeit not the only, indicator for development (see also section 2.3, p. 52). ICTs are considered to have high potential value across all sectors, both in public and private organizations, from software businesses in urban centres to health delivery in rural areas. The question has become not whether but how ICTs can be beneficial (Walsham and Sahay, 2006).

The increasing importance of ICTs in developing countries is reflected for one thing in the appearance of dedicated scientific journals such as *Information Technology for Development*, *Information Technologies and International Development* and *Electronic Journal on Information Systems in Developing Countries*. Additionally, special issues have been published in premier IS journals such as *MIS Quarterly* with its June 2007 issue dedicated to IS in developing countries (Vol. 31, No. 2), *Information Society* with its special issue on ICTs in developing countries in 2002 (Vol. 18, No. 2), and the upcoming special issue on theorising development and technological change in the *Information Systems Journal*.

Heeks (2008) provides several arguments in favor of the application of ICTs in the context of developing countries. At the macro level, Heeks argues that life is becoming increasingly digital, not only in more advanced nations. At the micro level it can be observed that poor communities already sometimes prefer ICT over alternatives to spend the little money available. Heeks also puts a moral argument forward of engaging with the world's major problems; most ICT professionals spend their lives serving the wealthier, already relatively well performing corporations of the world, but not to improve livelihoods of a large number of disadvantaged people, for who ICT could have far reaching impact. A further argument is that of *enlightened self interest*: In the contemporary globalized world, the problems of poor people of today can become the problems of others tomorrow, for example through migration or conflicts. Also, people in developing countries are potential new customers and trade partners. Finally, Heeks argues that working with African or Asian communities offers potentially interesting, rich, satisfying and colorful experiences. From a research perspective, there are further strong points in favor of investigating ICT initiatives in the development context: In development countries, the gaps between intended design and actual reality are often more explicit, thus making such projects extreme cases. ICT cases in developing countries help illuminate IS failure and underlying processes Heeks (2002b). The involvement of geographically and culturally distant partners from both industrialized and developing countries in a project makes the differences between contexts even more explicit (Heeks, 2001).

1.2 The problem of unsuccessful IS projects

Failure is a familiar theme in IS research. An IS project may fail to be completed, fail to produce expected results, or an IS may not be used at all (Avgerou, 2008). Failure is a problem for IS projects around the world, but it is especially severe in developing country contexts. A contributing factor to this situation is that in many developing countries, particularly in Africa, local information system development capacity is weak (Braa, Monteiro, and Sahay, 2004).

It is difficult to assess the amount of information system failure in developing countries due to limited available data, but there is some evidence that failure rates are considerably higher than in industrialized countries. Information system failure is a real problem for developing countries for several reasons, including the absence of learning from

failure and the high opportunity costs because of the limited availability of capital and skilled labour (Heeks, 2002b).

It is not possible to list all possible failure sources, but many fall into one of three kinds: organizational issues, system development issues, and management issues. Specific problems that often occur in developing countries include scalability failure, sustainability failure and assimilation in dysfunctional organizational processes (Avgerou, 2008). Scalability is about rolling out small scale pilot projects to become fully operational IS, or improving complexity of services over time. IS sustainability, which is often related to scalability, is about the tendency of systems to endure over time and space (Kimaro and Nhampossa, 2004; Walsham and Sahay, 2006).

1.2.1 Projects and (un)sustainability

The strong reliance of developing country based IS projects on donor funds presents a structural problem for sustainability. Donor projects may well have the intention to leave a maximum level of local improvements behind, but there are certain impediments. First, there is a contradiction of terms, because projects are by definition not sustainable, since they are defined investments with start and end dates (Young and Hampshire, 2000). Furthermore, donor funded projects are typically short term in nature (Kimaro and Nhampossa, 2004), having a life span of not more than a few years. This may be enough for short-term operationalisation. But it hardly suffices to prepare the ground to achieve long-term goals. On the contrary, IS adaptations that are useful in the short-term for initial installations, may undermine long-term IS institutionalization. For example, there is often a focus on the quantity of technical features, and at the same time a lack of institutionalization. This achieves allegedly impressive project results, but doesn't prepare the IS user organization adequately for the long term. In such cases, even if donor projects appear successful within their time frame, soon afterwards problems may surface that are rooted in insufficient considerations of long-term aspects. After the end of donor funded projects, it is not uncommon that user organizations find themselves confronted with ineffectively designed systems, and with insufficient human resource capacity to adapt and extend the systems in order to make them effective (Kimaro and Nhampossa, 2004). Consequently, in too many cases IS are soon abandoned.

Findings in other areas of development cooperation, outside the realm of ICTs,

are also relevant with respect to project sustainability. Commonly observed obstacles to project sustainability include (Sangmeister, 1998, citing Stockmann, 1996):

- Formulation of project targets without strong involvement of local partners;
- Negligence of training and education, contrasted by strong support for technical equipment;
- Lack of flexibility during project planning and implementation due to excessive focus on rigidity and goal-orientation.

1.2.2 Need for DCs to become active IS developers (producers)

A major difficulty of IS user organizations lies with long-term IS institutionalization. This is linked to a separation of the IS design and IS user realms. This separation is enormous when IS users are located in developing countries and IS developers as far away as in industrialized countries.

Despite the importance to overcome this separation for long-term sustainability, software development is not yet recognized widely in developing country contexts as a way of solving local problems. In a few countries, such as in India, software development has become more prominent, but often the focus is on export rather than on achieving local social and community objectives (Ezer, 2006). Despite some exemplary efforts to facilitate local system development (Keats, 2007; Korpela, Mursu, and Soriyan, 2002), many more projects have been undertaken by transferring technology from industrialized countries to developing countries (Macome, 2008) – with a visible interest in open source software (van Reijswoud and Mulo, 2006). However, many consider foreign technology import on its own as not being optimal for local socio-economic development. For example, the African Information Society Initiative states that "Africa needs to enter into the information age as a developer, not as a consumer", that Africa needs to become "less dependent on outside software producers and developers" and that Africans have the "right to make [their] own choice" (UNECA, 2010, p. 3). Similarly, (Gurstein, 2003) argues that for development to occur and to overcome the digital divide, individuals and communities need to become producers, not only consumers, so that they can make *effective use* of ICTs. This implies developing a strong "local skills base" (NACI, 2002, p. 15) and supportive organizational structures. The need to nurture conditions for local solutions is underlined by the following statement: "As the number of ICT projects in Africa increases, the skills of the internal

workforce to implement all these projects become less and less sufficient. This calls for the development of strong skills internally, instead of a dominant external consultancy" (Massingue, 2003, proposition 4). These perspectives are indications for both the relevance and the intention of becoming active participants in the information society, of becoming independent enough from foreign technology to take control of the future and to be able to solve local issues.

Solutions of technology related problems can be rooted anywhere in the spectrum between *make* and *buy* (Baark and Heeks, 1999). Buying off-the-shelf packages can be a good choice if the required functionality is covered by existing packages, such as in the case of operating systems and office applications. If there are no appropriate packages available that cover full or part of the requirements, then some local software development may be inevitable in order to achieve the intended goals. Local software development then can range from a few adaptations of existing software to full system development, either within a single organization or in collaborations between several actors. Baark and Heeks (1999) distinguish seven levels of technological capabilities (see table 1.1). The scale ranges from level 1, *non-production operational capabilities*, to level 7, *innovative production*. Baark and Heeks (1999) evaluated four Chinese technology projects and observed that at best, local developments lay around level 5, *minor production modification*.

There are strong roadblocks against local software development efforts. Standard software packages are often cheaper and of higher quality than local development can achieve, despite the potential of low-cost labor. This situation is intensified by widespread piracy. In addition, there is often a preference for foreign software. In sum, there is a domination of imported packages in many application areas (Heeks, 1999). But not all types of software systems are as readily available, as illustrated by custom applications and appropriate organizational information systems. Moreover, systems often need to be adapted and extended. The absence of local software development activities then prevents local communities and organizations to progress and resolve pressing issues. Not least, local context has to be taken into account (Avgerou, 2001), which limits the appropriateness of commercial off-the-shelf packages and methodologies. These limitations are increasingly being recognized in the context of developing countries (van Reijswoud, 2009).

The lack of active IS development can be seen as a lack of self-determination: "In its fullest sense, empowerment is a process that enables disadvantaged people to increase control over events that determine their lives. Empowerment cannot be given or taught but

Table 1.1: Scale of general technological capability (Baark and Heeks, 1999)

Level 1. Non-production operational capabilities	1a: Using the technology 1b: Choosing the technology 1c: Training others to use the technology
Level 2: Non-production technical capabilities	2a: Installing and troubleshooting the technology
Level 3: Adaptation without production	3a: Modifying the finished product to meet local consumer needs
Level 4: Basic production	4a: Copying technology 4b: Assembling technology 4c: Full production using existing products and processes
Level 5: Minor production modification	5a: Modifying the product during production to meet consumer needs 5b: Modifying the production process to meet consumer needs
Level 6: Production redesign	6a: Redesigning the product and production process to meet local consumer needs 6b: Redesigning the product and production process to meet regional/global consumer needs
Level 7: Innovative production	7a: Developing a new product to meet local consumer needs 7b: Developing a new product to meet regional/global consumer needs 7c: Developing a new production process 7d: Transferring a production process to other producers

is something people do for themselves" (Hecker, 1997, p. 784). This summarizes the problem of being trapped in the passive consumer role. There is a need for local participants to engage actively in information system development and to take ownership of these processes. Only by becoming more active producers is it possible to improve the success rate of IS projects.

1.2.3 IS research relevance and contextually oriented research

The practical relevance of information systems research has been to a large extent disappointing. A review of research articles in two leading information system journals concluded the lack of a systematic body of practical implications from IS research articles (Iivari, Hirschheim, and Klein, 2004). The predominant IS research philosophy over the last decades has been of the form *theory-with-practical-implications*, that is, the intention to develop a cumulative, theory-based knowledge body with the aim to make prescriptions (Iivari, 2007). In accordance with this philosophy, research achieved high levels of rigor, but its results were of limited practical relevance; managers were unable to understand research papers, and research results were unteachable to students. On the other hand, practically oriented research often lacked rigor. IS research was trapped in a dilemma of rigor versus relevance (Lee, 2000).

Information systems are human activity systems that are usually technologically enabled. The "essence of IS lies in the contextualization of the machine in the social system" (McKay and Marshall, 2005, p. 3). Therefore, IS is a socio-technical discipline, and it is not appropriate to subdivide IS into two separate components. This implies that the context of design and use is critical, and that research practices need to embrace such a worldview. The context of use must inform the design, and thus must be part of any evaluation of the technical artifact or the process by which the artifact was built (McKay and Marshall, 2005). When intervening in a system, the whole system needs to be considered. In the case of socio-technical systems, this means that the technological subsystem cannot be optimized in isolation. Such attempts will not only lead to suboptimal solutions, but can even lead to an infeasible outcome (Lee, 2000). If information systems are considered as socio-technical systems, then the relevance of IS research is dependent on its context orientation; by taking into account the social and organizational context, by evaluating what works in practice, IS research has the potential to improve its practical relevance.

1.3 Context

The context of the Mozambican Higher Education sector is the basis and motivation for subsequent theoretical considerations. This section first describes recent developments in the sector, as well as associated difficulties and needs that led to the development of an academic registry information system (ARIS), also called student information system (SIS), within a north-south development cooperation project. This is followed by a brief description how the development cooperation project has been realized.

1.3.1 The Higher Education context in Mozambique

Mozambique applied a policy that allowed a large number of universities to emerge, with more than 20 public and private universities operating at the time of writing. This development started in the early 1990s, related to the peace negotiations to end the civil war. Traditionally, higher education was only available in the country's capital, which is located at its southern tip, close to the border with South Africa. One of the conditions for peace was the provision of university education to the central and northern parts of Mozambique. A concrete result of the peace negotiations was the foundation of the Universidade Católica de Moçambique (UCM) by the Catholic Church in 1995. Subsequently, other universities opened throughout the country, one of them Universidade Mussa Bin Bique (UMBB), a Muslim university in the north of Mozambique.

In recent years the number of students rose quickly in Mozambique. Table 1.2 summarizes available statistical data from the Ministry of Education and Culture. The increased demand for study opportunities triggered an expansion of existing and the formation of new universities, both in the public and the private sector. The higher education institutions expanded outside the capital Maputo to all provinces of the country. This has created universities with faculties in different cities, creating institutions spread over a large geographical area. These developments pose challenges for the management and administration in the higher education sector.

Until a few years ago it was still feasible to manage student enrollments, marks and certificates with pencil, paper and spreadsheets, but these techniques do not scale well to current, and still rising, student numbers. They have proved to be labor-intensive and error-prone with large student numbers. Academic registrars often used to know most students at their faculty and remembered details about the students' study careers, which

Table 1.2: Development of student numbers in public and private universities in Mozambique (Ministério de Educação e Cultura, 2005, 2006)

Year	2003	2004	2005	2006
Public sector	11235	15113	18863	32081
Private sector	5990	7143	9435	11152
Total	17225	22256	28298	43233

helped in the daily administrative work. As this familiarity was lost due to higher student numbers, paper and spreadsheet based administration resulted in more difficulties for the administrative staff.

Another problematic issue is the need for improved data security. This corresponds to the danger of data loss as well as to potential fraud when data about students and their marks is not properly protected. Both dangers are substantial; the latter is aggravated by low salary levels.

The difficulties encountered by Mozambican universities in student administration has another problematic consequence: the timely compilation and delivery of correct statistics to the Ministry of Education, which requires these statistics for internal planning.

1.3.2 OPUS project history and personal involvement

My own involvement had its roots in a position as a development consultant, sent by the Austrian NGO called *HORIZONT3000*, to the UCM in the form of a "long-lived relationship between consultant and partner organization" (Bass, 2009). The role was that of an adviser to introduce information systems in the administrative domain at the university. The university had grown in the number of staff and students, and in this situation the university management considered information systems relevant, particularly in the domains of accounting and student records.

Other universities in Mozambique had similar interests in finding better ways of managing their student records. This was recognized by the Mozambican Ministry of Education and Culture (MEC). The Ministry itself had an interest in improved student records management at universities in order to receive statistical information faster and in better quality. This potential win-win situation resulted int the formulation and negotiation of a development intervention project between MEC and Nuffic, the Dutch government-

sponsored organization supporting higher education in developing countries. The project was carried out between 2005 and 2009. Locally unavailable expertise was provided by the computing centre of the Radboud university Nijmegen university in the Netherlands.

UCM, UMBB and three other Mozambican universities, representing various backgrounds and cultures, were invited to participate in the so-called OPUS[1] project, with the aim to design and install an academic registry information system (ARIS). This project promised to be a useful intervention, since it sought to address a clear need by all five participating universities, as none of them managed student records effectively. External input was welcome, because local IS project experience was limited. On the one hand, the project progressed well by producing incremental versions of the software and evaluating them together with the intended users during joint workshops. On the other hand, it eventually became apparent that even though local participants were happy to be part of the project, their involvement outside organized meetings remained low, e. g. when they were asked for certain contributions between meetings.

The Dutch university, which was entrusted with leading the process of delivering an appropriate IS, started this process with collaborative requirements workshops with Mozambican participants from universities and the Ministry of Education. An effort was made to find and apply existing systems to meet the stated requirements. But no system could be identified as suitable for the Mozambican context. Hence, the decision was made to develop a software system from scratch. This implied the need for good local software development skills for the long-term maintenance of the system, which added another level of complexity and thereby a particular challenge concerning sustainability.

During IS development, the long distance between northern designers and southern participants limited the amount of face-to-face interaction. Personal communication was limited to joint workshops in Mozambique, which were opportunities to test and discuss increments of the emerging system. Several attempts were made to integrate local users and developers at times between joint workshops, such as the design of automated report templates, or simply further testing of the current state of the system. Despite a perceived enthusiasm of local participants at the workshops, and their repeated affirmation that this is an important project for local universities, there was very little activity by local participants between workshops.

[1]The name e-SURA (electrónico - Sistema Universitário de Registo Acadêmico) was given by the Mozambican participants to the resulting software application for the use at Mozambican universities.

The observation and the unfolding of these problematic phenomena, and a lack of literature that would advise on how to deal with such a globally distributed project, triggered an accompanying research endeavor. The practical relevance of the accompanying research project for the Mozambican OPUS project was naturally considered important. At this time a trusted relationship had already been established between the UCM, the Dutch partners and me, and we sought continued cooperation regarding the OPUS project.

1.3.3 Scope and participants

The scope of the research covers the complete software development life cycle, including production and use of the student information system in Mozambique. The observed phenomena in the early phases of the OPUS project suggested a closer look at the following topics, in order to improve project success and sustainability:

- collaborative, distributed software development
- adaptation of the system to the particular needs of different universities (with a focus at the UCM)
- implementation of the system to make it useful to local universities
- capacity building of local users and technicians
- modeling of a long-term support structure

In the Mozambican project the set of participants comprises the Ministry of Education as the local coordination unit, the computing centre of the Dutch university of Nijmegen as the northern development partner, five Mozambican universities as local IS user organizations, and the envisaged Mozambican support center. The five invited Mozambican universities that benefited directly from the development cooperation project are:

- Universidade Eduardo Mondlane (UEM), Maputo
- Universidade Pedagógica (UP), Maputo
- Universidade Católica de Moçambique (UCM), Beira
- Universidade Politécnica (ISPU), Quelimane
- Universidade Mussa Bin Bique (UMBB), Nampula

Based on this outline of participants, a generalized collaborative framework is derived, which is shown in figure 1.1. In the implementation of an information system in the context of a development cooperation project there are several partners who work together. No single organization has the capacity to fully develop the desired information

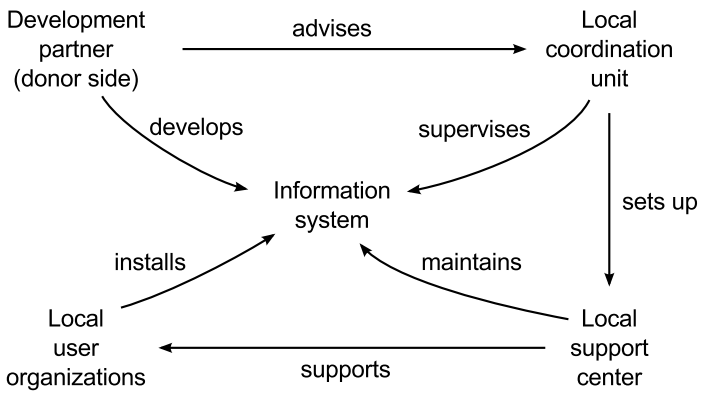

Figure 1.1: Collaborators abstracted from the Mozambican eSURA project.

system on its own. Software development can be done on the donor side as well as on the local side, possibly with the donor side starting with its high expertise and involving the local stakeholders increasingly as the project unfolds. Over time, local participants may be able to take full control of further maintenance. Typically, a local coordination unit coordinates between the local IS implementers and the donor side. Local support can be organized by the establishment of a local support center, which also can take over the further maintenance of the technical aspects of the information system.

Several levels of actors can be distinguished: on the personal level, local users of the system need to be addressed – their acceptance of the new system needs to be sought. Within the university certain preconditions should be met with respect to staff availability. The national level plays an important role for sustainability by setting up a support center. The development cooperation at the international level plays a major role in designing the system in a way that will be useful for the Mozambican universities and guiding the other actors based on available expertise.

1.4 Research objectives

The argument for the definition of the research objectives goes as follows. The starting point is the existence of one or more organizations with a need for a certain information system, but with limited experience in information system production and use. Because of the limited experience, external partners are required to address the local needs. Therefore, a network of at least one local and one external organization is required. The

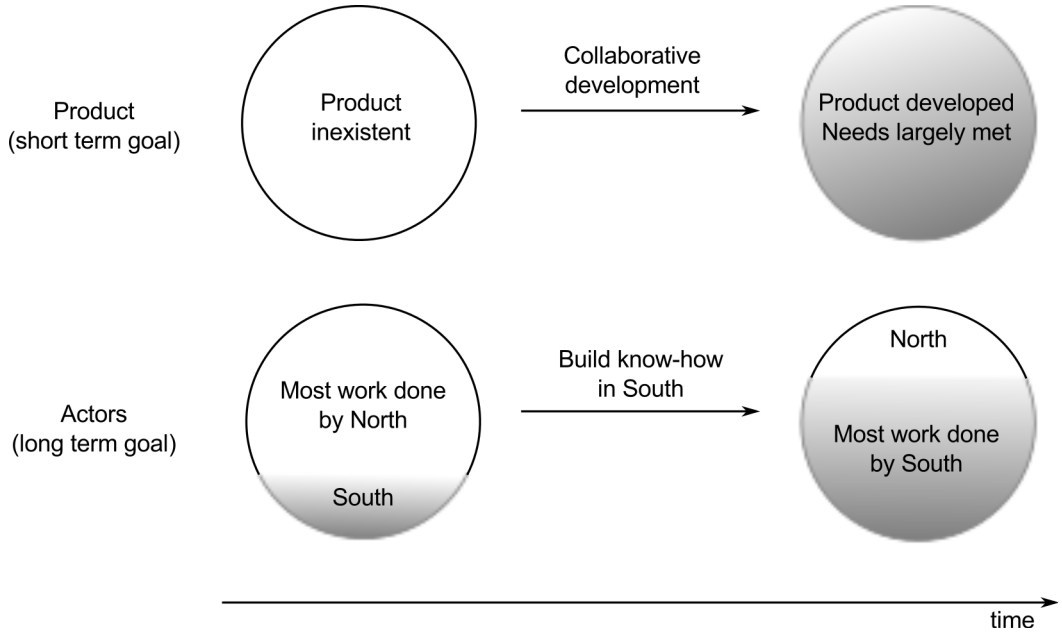

Figure 1.2: Two goals: creation of the information system (product) and of relevant skills (educational)

more general sense consists of a network of several local and several external organizations. In fact, reuse of the information system artifact in different user organizations is desired, due to financial constraints. The goal of the network is twofold, as illustrated in figure 1.2. First, to address the local organizations' information system needs in the short term, technical artifacts such as software and hardware have to be produced and organizational processes have to be adapted. To sustainably meet the information system needs in the long term, an appropriate infrastructure needs to be built that enables the support of the information system at all levels. This infrastructure includes human skills for production and use, technical infrastructure and the organizational environment. Users may need support from outside their own organization, i. e. somebody to call if problems arise. Therefore, required skills may be distributed across different organizations, including support organizations.

The argument leads to the formulation of research objectives. The main objective of the research is to investigate effective ways that contribute to the production and use of information systems in the context of scarce resources typically found in developing countries. The research is based on these specific areas of investigation:

1. Sustainability of cross-cultural information system projects
2. Information system support structure

3. Structure for putting an information system into the open source tradition
4. Proof of concept

First, an important objective is to find ways to avoid the common pitfalls concerning IS sustainability in information system projects. This is to be formulated in a way to make insights from the particular OPUS information system project useful for similar projects. Second, user support is expected to be an important factor for the sustainability of an IS, particularly in the case of donor projects with the abrupt termination of external input. The third objective is to work out a concrete setup of how the system can be put under an open source license, in order to enable collaboration. Fourth, the findings of the research shall be validated practically in the OPUS project.

The research objectives are rooted in practical needs in the realm of the Mozambican OPUS project, for which, according to the literature review, there is yet insufficient evidence that could prescribe best practice approaches. The aim of this research is to contribute to successful Information System (IS) projects both theoretically and practically.

1.5 Research questions

The research is concerned with the improvement of success and sustainability of information system projects in a cross-cultural setting. Thereby, the entire system life cycle is addressed. The overall question is stated as follows:

Overall research question. *How can the success and sustainability of cross-cultural information system development projects be improved?*

The overall research question can be broken down in several questions that concern particular aspects as follows. See the subsequent section 1.6 on the relationship between the research questions and different areas of information systems research.

Research question 1. *How can information system development projects deliver meaningful solutions to local problems? In other words, how can the relevance of information system development projects be ensured?*

Research question 2. *How can information systems be developed cooperatively, given the globally distributed character of the participants?*

Table 1.3: Classification of the research questions according to the five areas of information systems research of Avgerou (2000a)

1. Applications of IT to support the functioning of an organization	—
2. The process of systems development	RQ1 (Relevance), RQ2 (Open source)
3. Information systems management	RQ3 (Change), RQ4 (Support)
4. The organizational value of IS	RQ5 (Success)
5. The societal impact of IS	—

Research question 3. *How can local innovation be nurtured, so that local participants take active ownership and control in shaping solutions to local problems, rather than waiting for solutions to be delivered from outside?*

Research question 4. *How can information systems be supported in the long term, so that installed systems do not get abandoned despite their potential to solve local problems?*

Research question 5. *What are the relevant indicators of success and sustainability in cross-cultural information system project settings?*

1.6 Significance of the study

Sustainability has been characterized as an "important, but neglected topic" for IS in developing countries by Walsham and Sahay (2006, p. 16). Relatively little has been written to date on this topic. Open source software is characterized by Walsham and Sahay (2006) as a key issue where more research is needed, because open source is a potential enabler of cheaper and better ICTs for developing countries.

Avgerou (2000a) distinguishes a set of broadly five different areas of information systems research. Table 1.3 puts these five areas in relation to the research questions. The first of the five areas is application specific. It deals with applications like database technology, expert systems or decision support systems. Although the OPUS project is about building a specific application – a student information system – the focus of the thesis is less on the particularities of the application type of student information systems than on the process of cross-cultural information system production and use. That is, the

research intends to gain insights that are also valid for other types of information systems.

The second area, system development, has two aspects: for one, it is concerned with the engineering side of developing an optimal solution for the given technical and economic conditions. On the social side it is concerned with developing people's capacity to deal with the new technology and information that is made available in an organization. Research question 1 intends to find out how to make the system development process relevant to the local conditions and needs, from the technical point of view as well as the building of required capacity. Research question 2 looks at ways how to coordinate the system development process in a distributed fashion.

The third area, information systems management, is about issues like the alignment of information system development with business objectives, or using IT to achieve desirable organizational goals. In this respect, research question 3 looks for ways to root the system development process in the local context. A main concern is to involve all stakeholders in the development process, and to drive organizational change in coordination with the other development activities. Research question 4 looks at the wider picture of the network of user and support organizations. The question is how to organize support so that users aren't left alone and that they acquire the skills to help themselves as much as possible.

The fourth area, the organizational value of information systems, is associated with evaluation. Cost-benefit analyses are popular to assess IT projects. But for anything other than the simplest IT projects, not all benefits are easily quantifiable. Where organizational change is involved, assessment needs to go beyond the economic value of IT. For that, theoretical perspectives from the social sciences can be useful. Research question 5 asks for indicators for success in a broad sense, how the OPUS project is considered valuable for its stakeholders.

Area five, the societal impact of information systems, studies the impact of new technologies on wealth creation, working life social life more generally. There isn't any research question dedicated to this area. But as the introduction already stressed in section 1.2.2, there is an assumption that organizations in developing countries have no choice than to become more active producers of information technology, to take ownership of the solutions to the local needs. The research in this thesis attempts to contribute to such a development, which potentially affects wider issues, even without specifically asking research questions about this area of research.

1.7 Structure of the thesis

This introduction is followed by a chapter containing a literature review of research areas that are relevant to the thesis topic (chapter 2). The focus of the literature review lies with sustainability, which is viewed from several angles. Subsequently, possible research approaches are reviewed, and the research methodology for this thesis is described (chapter 3).

In part two, a series of chapters address particular perspectives related to the research objectives. This includes an investigation of collaborative IS development from the perspective of *appropriate technology*, which has a long history of guiding technology application in developing country contexts (chapter 4). This assessment provides an orientation for subsequent chapters to build upon. The following chapter looks at opportunities of open source for collaborative IS development and proposes an open source structure for north-south collaboration (chapter 5). This is followed by a chapter on local capacity building within the institution, with the aim of minimizing the distance between users and designers (chapter 6). Thereafter, external support is being investigated (chapter 7). After the different perspectives have been worked out, a structural analysis of the complete Mozambican OPUS project gives an insight into success as perceived by the various project stakeholders (chapter 8).

Then, the main result is formulated in chapter 9 in the form of a methodology. The Appropriate Information System Development (AISD) methodology integrates findings from previous chapters and presents a set of theorems. Finally, the conclusion presents a design theory that incorporates the findings in previous chapters.

There are several possible reading paths through this thesis. Chapters 4 to 8 provide the steps that allow to formulate the appropriate IS development methodology in chapter 9. Alternatively, one may start with chapter 9 to get an overview of the AISD methodology, and follow the references to other chapters for further details. Also, some readers may be interested specifically in individual chapters.

1.8 Paper contributions

A short overview is given of the main contributions of the already published papers that form the basis of this thesis. With the exception of the research proposal (chapter 3)

all papers were peer-reviewed. Concerning author contributions, most work was done by myself, but valuable input was also given by the stated co-authors, by making comments during paper writing.

Chapters 1 and 2

Markus Pscheidt and Theo van der Weide (2010b). "Sustainability of collaborative information system development projects – a North-South case study." In: *4th IDIA Conference: Exploring Success and Failure in Development Informatics: Innovation, Research and Practice*. Cape Town, South Africa

This paper motivates the need that organizations in developing countries shall move away from a predominant consumer-orientation and engage in becoming active producers of software and information systems. Furthermore, literature on sustainability and success in the realm of information systems is reviewed in order to clarify the terms. Finally, based on a case study on the OPUS project, a set of approaches are identified to improve north-south project sustainability in three main categories of utility (social sustainability), embedding (political sustainability) and capacity (Skills, financial and other resources).

Chapter 3

Markus Pscheidt (2008). "Sustainability Factors for Information Systems in Developing Countries – Academic Registry Information System in Mozambique." In: *Prato Community CIRN Conference 2008: ICTs for Social Inclusion: What is the Reality?* Prato, Italy

This was written as the research proposal and presented at the Prato conference in the PhD student stream. In contrast to the other publications it was not peer-reviewed. It outlines the research approach for the research project, proposing action research and design science research. The overall research objective is established as improving the poor sustainability of IS projects in developing countries. To achieve this, several key aspects are identified: Solutions must be rooted in local needs; cooperation between organizations is required, because single organizations typically do not have the means to solve their information system needs on their own; organizational conditions for the implementation of information systems and the associated organizational change; and the need to support user organizations. These aspects are expected to contribute to the overall goal of information

system success and sustainability.

Chapter 4

Markus Pscheidt, Victor van Reijswoud, and Theo van der Weide (2009). "Assessing Appropriate ICT with ARIS case in Mozambique." In: *ICCIR'09: 5th Annual International Conference on Computing and ICT Research*. Makerere University. Kampala, Uganda

The paper on appropriate technology made observations about the specific characteristics of information system development in relation to appropriate technology principles, such as the orientation on local needs, the use of locally available materials, or the maintainability of technical solutions by the local community. One of the co-authors, Victor van Reijswoud, had already applied appropriate technology principles to the realm of ICT projects. ICT projects are however more general than IS projects; IS are evolving through changing requirements over time. This paper's contributions lie in conclusions about information system development in relation to appropriate technology. This includes a way forward towards an overall IS development methodology and the integration of artifacts into the system development life cycle.

Chapters 4 and 5

Markus Pscheidt and Theo van der Weide (2010a). "Bridging the Digital Divide by Open Source." In: *International Journal of Innovation in the Digital Economy, Special Issue on Digital Divide* 1.2, pp. 44–60

This journal publication brings together results from the papers on appropriate technology (Pscheidt, van Reijswoud, and van der Weide, 2009) and open source information system development (Pscheidt, Simons, and van der Weide, 2009). It emphasizes how such an approach can put less experienced organizations in scarce resource environments in a better position to make effective use of information technology by better taking control of local problem solving, in contrast to the widespread import of information technology from abroad, which creates dependencies on foreign technology providers.

Chapters 5 and 6

Markus Pscheidt, Eduard J. Simons, and Theo van der Weide (2009). "Towards 'best practices' in North-South Open Source projects – lessons learned from the ARIS project in Mozambique." In: *3rd IDIA Conference: Digitally Empowering Communities: Learning from Development Informatics Practice*. Berg-en-dal, Kruger, South Africa

A possible scenario is elaborated for collaborative information system development between northern and southern partners. The suggested scenario is based on open source software development. It outlines the roles of participants with different levels of expertise and a possibly existing donor. It identifies prerequisites for globally distributed software development collaboration, such as a modular system structure. Building blocks are derived for the different phases in the system development life cycle. The building blocks were worked out together with Ed Simons. They include the balancing of community trust and economic opportunities with a proper license choice; spin-out of a working system; the increasing involvement of local organizations into system development; and the reservation of development resources for the time of implementation. These paper contributions form the basis for chapter 5.

Additionally, a set of roles is identified that together form a *stable project structure* for the implementation of collaborative information system development. This aspect was inspired by Theo van der Weide. It has been integrated into chapter 6.

Chapter 7

Markus Pscheidt and Theo van der Weide (2009). "Supporting the ARIS community system in Mozambique." In: *Prato Community CIRN Conference 2009: Empowering communities: learning from community informatics practice*. Prato, Italy

If organizations are not supported after technical solutions were introduced, the risk is high for the new technologies being abandoned. Therefore, a multi-level support structure is proposed in this paper. Support activities are rooted in the empowerment of less-experienced organizations. Empowerment is considered from a community development perspective. The paper proposes IS support activities in four different categories: service delivery, capacity building, advocacy, and social mobilization. Service delivery includes service activities such as IS installation or software feature development. Capacity building includes e. g. a variety of training, such as workplace based training. Advocacy tries to

make sure that the voice of users are heard, e. g. through feedback or proper representation. Social mobilization includes bringing together organizational actors and technology to drive organizational change.

Chapter 8

Markus Pscheidt (2011). "Structurational analysis of cross-cultural development of an academic registry information system in Mozambique." In: *Information Technology for Development* 17.3, pp. 168–186

This paper looks at the entire Mozambican OPUS project by analyzing the differences in success measures between different participants of the project. It does this by taking a closer look at cultural differences between participants. Although the Hofstede model is frequently used to analyze cultural differences between nations, a different approach is followed. The structuration theory by Giddens, which was adapted by Walsham for the domain of cross-cultural information system projects, is used to analyze potential and actual conflict between participants. This theory allowed to carry out the analysis of the OPUS project (a) on three levels: the global project level, the Mozambican inter-university level, and the individual and group level within one university, and (b) with a temporal dimension, by analyzing how events unfolded over time. The paper's contribution includes a set of five success dimensions that are seen as relevant for such cross-cultural projects. Some of the success dimensions are specifically concerned with the aspect of sustainability.

Chapter 2

Literature review

After the introduction has set the motivation for the investigation of information system success and sustainability, this chapter will provide a review of relevant literature for the research project. First, a short overview is given of sustainability in general. Subsequent sections will then further look at the issue of sustainability in the areas of information systems, developing studies and computer science. Each of these fields approaches sustainability from a different angle, which will be integrated in later chapters.

2.1 Background and history of sustainability thinking

The Brundtlandt report (Brundtland, 1987) had a strong impact on raising awareness of the importance of sustainability for human development. It is often referenced regarding its definition of sustainable development, which is characterized as a form of development that meets the needs of the present without compromising the ability of future generations to meet their own needs. At the UN conference on environment and development in Rio de Janeiro 1992, sustainable development has been formulated as a fundamental goal for the future. The so-called *Rio Declaration* lists a set of principles to guide sustainable development, which span all political fields. Sustainability has not only an ecological, but also social and political dimensions, since it implicates social justice and democracy.

Sustainability is also relevant to projects and programs in developing country contexts. On the one hand, ICT4D solutions shall be "complementary to sustainable development" (Gómez, Martínez, and Reilly, 2001, p. 113). On the other hand, it is a challenge to keep ICT4D projects themselves in existence after funds for pilot projects have finished.

Isolated, small-scale projects have often been found to have limited impact and not being able to contribute to necessary local structural change. Thus, they often suffer from weak sustainability. To make projects and programs sustainable, for one thing they need to be able to diffuse sufficiently to make a broad impact. For another thing, they need to facilitate local innovation (Nuscheler, 2004, p. 473).

Although projects are often being monitored and evaluated during their implementation, there is a lack of evaluation concerning the time after donor withdrawal: how well projects operate, how effectively they are sustained, and to what extent they produce the intended effects (Bamberger, 1991; Stockmann, 2000). In other words, there is little evidence about the sustainability of projects.

2.1.1 Context-free sustainability concepts

Although sustainability has often been associated with environmental concerns, the ideas of sustainability and sustainable development have entered many fields, such as human societies, cultural traditions or social institutions. Sustainability is therefore an *umbrella concept* (Loukola and Kyllönen, 2005). Hence, it is necessary to investigate what sustainability means in a particular scientific field. A context independent statement about sustainability is the following:

> "The key component of the concept of sustainability is a requirement for the sustenance, survival, or flourishing of a process, an organism, or a resource. The viewpoint here is broader than usual: the entity to be sustained often consists of a large variety of interacting factors in a complicated setting" (Loukola and Kyllönen, 2005, p. 2).

Although there doesn't exist a unique definition of the term sustainability, it is often related to maintaining something that already exists over time in a self-sustaining and self-sufficient manner, without requiring outside support (Kimaro and Nhampossa, 2005). Key questions to categorize the philosophies of sustainability are: (a) What to sustain?, (b) Why should it be sustained?, and (c) How to sustain it? (Loukola and Kyllönen, 2005, p. 3)

For information systems, it isn't trivial to state the entity to be sustained (the *what*), nor *why* and *how* to sustain it. Information systems are more than technical artifacts, but also comprise the people working with them and the processed information; information

systems are socio-technical systems (Bostrom and Heinen, 1977). Sustainability in relation with information systems will be further analyzed throughout this chapter.

Common underlying characteristics of many streams of thought concerning sustainable development are (IISD, n.d.):

Concern for equity and fairness: Inclusion of less privileged and future stakeholders

Long-term view: Short-term view of project implementers versus long-term interests of beneficiaries

System thinking: Interrelations between all aspects of systems, and consequences of decisions

A further perspective is that of sustainability being a property arising out of the interactions among stakeholders; sustainability as something that is being negotiated (Batchelor and Norrish, 2003; Gumucio-Dagron, 2003).

2.1.2 Sustainability and organizations

As an intermediate step, some general observations can be drawn that are specific to organizations, but not specific to information system projects. This is relevant because information systems are often implemented in organizations. In organizations, sustainability is often conceptualized through *routinization*.

Whereas the primary sustainability process of routinization concerns the intra-organizational aspects of sustainability, in the wider ecosystem of inter-organizational relationships, there is a secondary sustainability process of *standardization*. Standards include institutional standards, rules and policies. Standardization represents a higher degree of sustainability than only routinization of activities. Based on routinization of activities and standardization, project sustainability can be classified in three distinct degrees (Pluye et al., 2004):

Weak sustainability: Non-routinized activities

Medium sustainability: Routinized activities (activities meeting all the characteristics of organizational routines)

High sustainability: Standardized routines (routinized activities complying with rules or policies)

Sustainability is often conceived as pertaining only to later project phases. In this view, sustainability is not addressed during the earlier phases of planning, design and im-

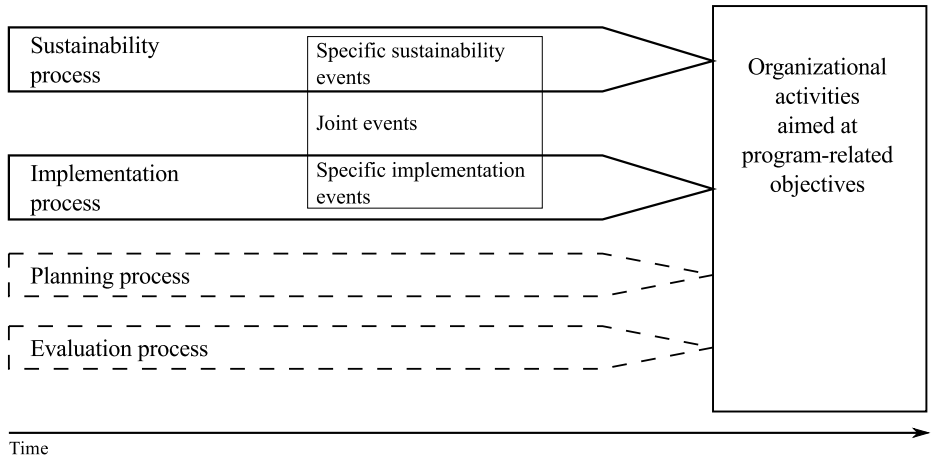

Figure 2.1: Concomitant sustainability (adapted from Pluye, Potvin, and Denis (2004))

plementation. In contrast, Pluye, Potvin, and Denis (2004) argue, based on public health programs experience, that sustainability needs to be prepared from the outset, meaning that sustainability shall be included in the planning process, and that sustainability activities be carried out concomitantly with other project related activities. This is illustrated in figure 2.1. Similar to the concomitance of project implementation and sustainability activities, it is argued that planning and evaluation processes cannot be neatly distinguished as separate phases in time. Planning does not stop when implementation begins, as well as evaluation is better considered a continuous process from the beginning through all phases of implementation. This challenges the *stage model*, which consists of distinct steps for planning, followed by implementation, and finally evaluation.

2.2 Information systems

This section starts from a general overview of IS research, and distinguishes the IS field from related fields. Subsequently, literature in the sub-field of IS in developing countries is reviewed. This is followed by a closer look at IS success and sustainability.

2.2.1 IS literature in general

The field of information systems investigates Information and Communication Technology (ICT) in organizational or community settings. It is an applied research discipline in which the "focus should be on how to best design IT artifacts and IS systems to

increase their compatibility, usefulness, ease of use or on how to best manage and support IT or IT-enabled business initiatives" (Benbasat and Zmud, 2003, pp. 191).

The IS field had some difficulties in defining itself and in marking its boundaries against engineering on one side and behavioral science on the other. The dilemma is that if a technology approach to IS is chosen, then there is no clear borderline to engineering. Similarly, if a behavioral approach to IS is chosen, then other behavioral disciplines could readily do IS research (Lee, 2000).

IS research frequently utilizes theory from other disciplines like computer science and social science to solve problems at the intersection of information technology and organizations (Pfeffers et al., 2007). For example, from the social sciences, a wide variety of theories has been borrowed by IS researchers (Orlikowski and Barley, 2001), such institutionalist theory (Avgerou, 2000b) and structuration theory (Giddens, 1984). In some cases additions to theories from other fields have been made by IS research. But the IS field has struggled to come forward with its own theories: "It is possible that the current emphasis with theories from other disciplines has distracted the IS research community from developing its own theories" (Benbasat and Zmud, 2003, p. 192). Due to the widespread reference to related disciplines, some have attributed them the status of *reference disciplines*. Others criticize this strong relationship as overemphasizing the importance of related disciplines; Lee (2001) argues that the interaction between social and technical systems, which is characteristic of IS, can only unsatisfactorily be described by models of related disciplines, because they are inevitably concerned with only a subset of phenomena of interest to the information systems field. Therefore he considers them as "contributing disciplines at best" (p. iii).

A characteristic that distinguishes IS from other fields is that it is concerned with the use of human-machine systems (Gregor and Jones, 2007). In other words, IS research is not just interested in the social system, or the technical system, or the two side by side. Of central interest in IS research are also the phenomena that emerge when social and technical systems interact (Lee, 2001). The relevance of the IS field at the intersection of social and technical approaches can be illustrated as follows: Methods for technical systems analysis, such as object oriented analysis, cover the *hard* aspects of the problem domain, but organizations are typically of an "ill-structured and fuzzy kind", because people are different in nature from data and processes. People have different and conflicting objectives and perceptions, and people change over time. In the process of introducing IS into

Table 2.1: Characteristics of tool view and socio-technical system view (Kling and Lamb, 1999)

Tool model	*Socio-technical model*
IT is a tool	IT is a socio-technical system
Business model is sufficient	Ecological view is needed
One shot implementation	Implementation is an ongoing social process
Technological effects are direct and immediate	Technological effects are indirect and involve different time scales
Incentives to change are unproblematic	Incentives may require restructuring (and may be in conflict with other organizational actions)
Politics are bad or irrelevant	Polities are central and even enabling
IT infrastructure is fully supportive. Systems have become user-friendly, people have become computer-literate, and these changes are accelerating with the "net-generation"	Articulation work is often needed to make IT work. Socio-technical support is critical to effective IT use.
Social relationships are easily reformed to take advantage of new conveniences, efficiencies and business value.	Relationships complex, negotiated, and multivalent.
Social effects of IT are big but isolated and benign	Potentially enormous social repercussions from IT
Contexts are simple (described by a few key terms or demographics)	Contexts are complex (matrices of businesses, services, people, technology history, location, etc.)
Knowledge and expertise are easily made explicit	Knowledge and expertise are inherently tacit/implicit
IT infrastructure is fully supportive	Articulation needed to make IT work

organizations the human aspects need to be addressed. "Failure to include human factors may explain some of the dissatisfaction with conventional information systems development methodologies; they do not address real organizations" (Avison et al., 1999, p. 95).

The application of IT into organizations should consequently not be approached simply as tool usage. System design inevitably encodes assumptions about the social organizations, for example in the definition of authorizations to execute certain information system features. Table 2.1 characterizes key differences between the tool view and the socio-technical system view. One of the consequences of following the tool view for IS implementation is underestimating the complexity and time that is required for organizational changes during IS implementation (Kling and Lamb, 1999).

Information systems are being applied to a wide range of application domains and contexts. IS development can be illustrated as an ongoing interaction between technical artifact development and the context for which IS artifacts are being designed:

"A perennially interesting research topic in the IS field is how to effectively

> develop new systems. The topic is interesting because, as IT develops and technical knowledge grows, IT is applied to new application areas that were not previously believed amenable to IT support. In the process, new kinds of systems and new development methods are also created" (Markus, Majchrzak, and Gasser, 2002, p. 180).

An important part of IS research is software engineering, which deals with all aspects of the software development process, including production, organizational implementation, evolution and evaluation. But there are also important differences between information systems and software engineering. Whereas software engineering is concerned with all types of software, such as scientific applications and embedded systems, IS research is focused on a particular type of application, information systems software, which is specifically concerned with aspects like transaction processing and decision support. The software engineering component in IS is concerned with IS development from an organizational perspective. Organizational factors such as requirements, processes and opinions play a major role. Such factors are relevant on the level of the individual, group or organization (Morrison and George, 1995).

The historic development of the IS field has been diverse in different regions of the world. In North America, the IS field was originally part of management science, and became an independent field in the 1960s. The predominant research approach has been behavioral research. After being increasingly under pressure for producing results with meager practical applicability (Benbasat and Zmud, 1999; Kock et al., 2002), one reaction was the appearance of interpretative studies. But the IS field continued in a state where some claimed it to be in crisis (Benbasat and Zmud, 2003). In recent years design science research has been suggested as an additional approach in order to improve the practical value of IS research. Design science is still in the process of becoming fully established in the IS discipline (Kuechler and Vaishnavi, 2008b). In Europe the preferred research approach varies geographically, but here design science has a stronger tradition. Especially in German speaking countries, where the IS field is called 'Wirtschaftsinformatik', design activities are well rooted. However, in many cases the design approach has not been made explicit. Within Europe the design science tradition can also be found in the Nordic countries, the Netherlands, Italy and France (Winter, 2008).

2.2.2 Information systems in developing countries

Avgerou (2008) characterizes IS research in developing countries as being distinct from mainstream IS research by its attention to the context of IS innovation and by discussing the developmental role of IS innovation. Therefore, IS research in developing countries should be able to make significant contributions to understanding the impact that historically constructed social conditions have on IS innovation, and to understand how IS interventions are able to improve life or working conditions.

Walsham and Sahay (2006) provide an overview of the current research on information systems in developing countries. Based on a grounded theory approach by analyzing a set of journal and conference papers, they developed a classification approach for literature in this area. This classification is used in the following to guide a structured overview of topics that are relevant to this thesis. This classification broadly distinguishes IS research contributions into the following categories:

1. Key challenges for the use of ICTs
2. The role of technology
3. Theory and methodology

According to Walsham and Sahay (2006), the role of development, what it means and how ICTs can promote it, is often implicit or underemphasized in a majority of reviewed studies, and for future research they call for more attention to the development to which ICTs contribute. Development can be viewed from many viewpoints. High level policy at government level is one area. Furthermore an increased focus on economic issues is important; whereas cultural issues are relatively often analyzed, they need to be more frequently accompanied by financial considerations in order to tackle financial sustainability of pilot projects. A further potential area of development is to improve freedom of choice and opportunity, as for example put forward by Sen (1999).

2.2.3 Key challenges for the use of ICTs

Key challenges range from broad societal or national issues down to the group and individual level, and have been identified as (a) how ICTs promote development, (b) promoting cross-cultural working, (c) local adaptation and cultivation, and (d) focusing on particular groups.

How can ICTs promote development?

Articles in this category deal with broad contributions of ICTs to development, sometimes in the context of a specific country. In some cases the meaning of the term development itself is discussed. Articles often draw on secondary data, in contrast to the more micro-level studies, which in the majority of the cases draw on primary data.

For example, (Madon, 2005) examines the use of the Internet in sectors such as health and education and in domains such as economic productivity and sustainable development. The conclusion is that the attainment of development goals does not depend on the existence of Internet connections alone, but on the acquisition, usage and dissemination of relevant information and knowledge.

This position is strengthened by the world development report (World Bank, 1999), which acknowledges that economies are built not merely through the accumulation of physical capital, but on a foundation of learning and adaptation. Furthermore, the report makes a distinction between technical knowledge, which guides how to carry out interventions, and knowledge about attributes such as the quality of a product or the credibility of a borrower. Accordingly, due to the unequal distribution of both types of knowledge, *knowledge gaps* and *information problems* are identified. The world development report concludes that the widening access to knowledge through the increased availability of ICT transforms relationships globally, for example between expert and amateur, or government and citizen. An important point to take away from the report is that knowledge cannot be static nor can it move only in one direction, but has to constantly flow back and forth across an ever-changing web of actors of those who create and use it.

Silva and Figueroa (2002) deal with a specific country, Chile, and discuss how to promote the improved use of ICTs in an institutional context, drawing from institutional theory, including particular adaptations of the theory to the IS field. In terms of technology they emphasize the importance of standards and telecommunications infrastructure in supporting ICT applications.

Sayed and Westrup (2003) address the question of how ICTs are mobilized to achieve networks that link global organizations, the government and local companies. The research is based on an example of the installation of an ERP system in an Egyptian company. They argue that ICT facilitated development has led to the formation of new, more complex networks of relations.

Promoting cross-cultural working

Myers and Tan (2003) note that much of the literature concerned with cultural and cross-cultural issues in the IS field has relied on the work of Hofstede (1980, 1991). They argue that the Hofstede's concept of national culture, which has tended to dominate culture related IS research, is too simplistic. They propose to IS researchers to adopt a more dynamic view of culture that is contested, temporal and emergent.

In the same direction goes Walsham (2002). Despite recognizing the importance of the work of Hofstede as a reminder of the importance of cultural difference, Walsham is critical in the sense that Hofstede's work is rather crude and simplistic for the study of cultural aspects related to IS. Walsham (2002) examines cross-cultural issues in software production and use with the goal of developing a theoretical basis for analysis in this area. The result is a model of the processes involved in IS production and use, drawing mainly from structuration theory.

Liu and Westrup (2003) examine coordination and control between UK and China in the context of a multinational corporation. In spite of the existence of e-mail and fax, ICT-enabled coordination is found only to be effective when linked with other approaches such as the use of expatriates and face-to-face contacts.

Braa, Monteiro, and Sahay (2004) look at sustainable health information systems in developing countries and more particularly in the case of the transfer of a district health information system from South Africa and Mozambique. Their focus is on actor networks involving both north-south and south-south relations. The experience from this extensive information system project suggests that the institutionalization process is strengthened when the network is extended and more groups of people align their interests with that of the IS.

Local adaptation and cultivation

Bringing a technology to a new local context involves elements of cultural transfer and mutual learning. Local knowledge is often required to be combined with the knowledge and technology being brought in from abroad.

Bada (2002) describes local adaptations in a case study about a Nigerian bank, which implemented a business process re-engineering project, as a form of IT-enabled change initiative. He emphasizes the importance of the local context of adapting IT-based practices

when implementing them in developing countries, arguing that globalization in general is perceived differently in different parts of the world, and that IT is associated with management solutions and organizational models and therefore calls for local customization.

Macome (2008) looks at the transfer of an invoice information system designed by a French company to the Mozambican electricity company EDM. Like Bada she concludes that the local context should be considered in the implementation of information systems designed for contexts different to the local implementation context. Furthermore she advocates the involvement of local stakeholders in the entire process.

Focusing on particular groups

Some studies have concentrated on certain groups that are suffering because they live outside the margins of the digital divide.

The study by Mosse and Sahay (2003) is located in Mozambique where the introduction of a health information system is supposed to improve the health provision to poor districts. They consider it important to build so-called *counter networks*, using a term by sociologist Manuel Castells. The role of communication is highlighted as playing an important basis in the strengthening of the network.

Some research has been carried out about research in developing countries itself, looking at difficulties for researchers to have access to scientific material. Ahmed (2007) assesses and evaluates the *Open Access* movement as a proposed solution to give better access to scientific resources to researchers in development countries. Open access is strongly linked with open source. Therefore, some of the problems of implementing open source have a direct impact on the implementation of open access. In order to tap the potential of open access for Africa, some key success factors are proposed, including the need to produce economic value within the region and the need for intensive training of users and developers of open access solutions.

Another study about access to knowledge sources was done by Okunoye and Karsten (2003), looking at how researchers in Namibia and the Gambia use traditional and Internet based communication technologies such as e-mail and web databases to manage knowledge. The analysis concerns the benefits and issues of the different technologies for knowledge creation, transfer and storage in the context of scarce resources. Although many benefits were identified in the use of the Internet, e. g. better access to research material

in comparison to what is typically offered in university libraries, a lot of issues hindered its effective use, including limited access, high cost and unfamiliarity with searching web-based databases. In some cases, undirected searches by inexperienced users resulted in the use of low quality references.

2.2.4 The role of technology

Technology is seldom discussed as a central issue in IS literature in developing countries. Studies that do discuss technology, into three distinct categories: (a) standardization versus localization, (b) alignment of actors in networks, and (c) particular technologies.

Standardization versus localization

Papers in this group deal with the conflict between global best practices and the difficulty of imposing these standards on different local contexts.

Braa and Hedberg (2002) describe the initial period of an action research study about the so-called Health Information System Project (HISP) to support health management at the district level in South Africa. The reconstruction of the health sector in post apartheid South Africa asked for a decentralized structure based on health districts. In terms of information systems this translates into a need to balance local control and project-wide standardization. The action research study involved the development of a modular hierarchy of standards to address tensions between standardization and localization. Therefore, the HISP software was designed with the ability to be tailored to local needs. Furthermore a *participatory prototyping* approach was chosen with frequent develop/deploy cycles, giving local stakeholders regularly the possibility to give feedback and adjust requirements. The software was released under an open source license. Parts of the health information system were translated into Portuguese and implemented in Mozambique.

M. P. Thompson (2002) did another action research study related to HISP. The study used an ethnographic style and focused on how individual users generate meaning and how this meaning generation is being used to align the information system with the user requirements. Even though many stakeholders considered the project successful, one particular obstacle remained: the continuing inability of some of the most disadvantaged clinics to collect and use health data which they found meaningful, which resulted in the submission of inaccurate, often meaningless data to the department. Thompson looks at the

generation of meaning at the individual level, observing that raw data is generated mostly through manual forms. He notes that difficulties arise due to a mismatch between the needs of the system and the local knowledge of the nurses and others who create the data, and argues for locally relevant data collection methods such as simple counters and forms.

Alignment of Actors in Networks

Some papers make use of actor-network theory (ANT), a relatively popular theory in IS literature as a way to conceptualize technology as one of the actors in actor-networks. Examples of papers which draw on ANT are the ones from Mosse and Sahay (2003) and Sayed and Westrup (2003). Braa and Hedberg (2002) refer to ANT, involving people, organizations, software and standards.

Rolland and Monteiro (2002) elaborate on the particular tensions between standardization and localization of an information system being implemented in a large distributed shipping company located in about 300 sites in 100 countries around the world. They argue that both standardization and local adaptations are necessary. The two opposing forces can only be managed by ongoing negotiations throughout the design process. They argue for a *reflexive design process* based upon iterative system design and ensuring flexibility to always keeping possibilities open for redesigning the system. Furthermore, information systems cannot be all-embracing perfect solutions, but shall rather be conceptualized as parts of larger information infrastructures with information and communication flows between the elements.

Particular technologies

Particular technologies include Internet, ERP, and free and open source software. Studies rarely go into great detail about the technologies, though. For example, Braa and Hedberg (2002) mention that their software has been designed under an open source license but do not provide further details.

Weerawarana and Weeratunge (2004) examine the open source phenomenon as it relates to developing countries. According to their assessment, open source software development offers "unprecedented opportunities" (p. 8) to position a local software industry. One of the key drivers for building up a local software industry is participation in global open source software development projects such as the Apache server. The authors however

warn that in the absence of appropriate leadership and human resources, the gap between strategy formulation and execution may be insurmountable. Therefore, organized frameworks are needed to guide the building of local open source software development capacity, rather than leaving this task to individual software developers on their own. Weerawarana and Weeratunga argue that open source based strategic initiatives have a good potential to "create value through key drivers of business opportunities, reduced investment cost and greater efficiency and effectiveness of government" (p. 5). They furthermore present possible developing country business models, a strategy framework of how developing countries can take advantage of open source and how a donor agency can assist towards exploiting open source to create value in the economy.

van Reijswoud and Mulo (2006) discuss the experience of a university in Eastern Africa in moving away from proprietary software and instead adopting open source software. Installation of open source software at the server side proved to be a big technical challenge, due to the absence of technical experts in the region. One of the strengths of open source software, increased choice of applications, has more than once turned into a disadvantage and led to confusion of users, e. g. by being overwhelmed by the many available Linux distributions. Top level management support was important, it however did not guarantee complete success. Furthermore, the authors consider continuous user information and education important, since resistance tended to build up quickly after presentations and workshops. One of the reasons for resistance is the assumption that open source software represents inferior software, for example with respect to office applications. This impression is reinforced by international organizations that do not use open source applications, and therefore do not provide role models. For technical implementers, appropriate support has been a point of concern; international mailing lists were only of limited help because questions posted were considered too basic by readers of the mailing lists in more industrialized contexts. The authors recommend the establishment of regional support centers for system administrators and users.

2.2.5 Theory and methodology

Theories and methodologies cannot be neatly grouped, but they are broadly distinguished into (a) existing theories that other scientific areas drawn upon, (b) new theories and concepts emerging from IS research, and (c) methodologies used. A wide range of

theories are being draw on by researchers of IS in developing countries, from fields such as globalization, development and sociology. Theories which derive directly from the IS field are less common (Walsham and Sahay, 2006). Some researchers have developed their own models, such as the design-reality gap model described by Heeks (2002b).

Many conceptual frameworks and normative models that guide IS practitioners are linked to *best practices* irrespective of the contextual particularities. Typically, IS innovation in developing countries involves the transfer of technologies and organizational practices that were designed in other socio-organizational contexts (Avgerou, 1995). Their value can however not be taken for granted. On the contrary, there is evidence indicating the significance of addressing the local context for the introduction of ICTs in developing countries (Avgerou, 2001).

An example of a model that takes into account local conditions is the *ITPOSMO* model (Heeks, 2002b). It intends to analyze gaps between design and reality in various dimensions, which serve as an indication for the feasibility of information system projects. Thereby, large design-reality gaps indicate potential difficulties during project implementation. The analysis may also be used to take appropriate steps to improve success rates by lowering design-reality gaps. ITPOSMO is a short form indicating seven dimensions in which gaps are analyzed for information system projects: Information; Technology; Processes; Objectives and values; Staffing and skills; Management systems and structures; Other resources.

The PADTR methodology was designed to guide the implementation of inter-organizational service systems in volatile environments. It focuses on the collaboration of public organizations in the developing world (Mulira, 2007). PADTR stands for *Preparation, Analysis, Development, Testing, and Realization* and is therefore a methodology that covers the whole software development process.

With respect to methodology, the majority of the studies claim to be interpretive. Few studies adopt a positivist approach with stated hypotheses, instruments for data collection, statistical inference etc. Many papers build upon in-depth case studies. Walsham and Sahay (2006) consider action as particularly relevant in addressing issues of development, but find that as of yet there are only few action research studies. One of the rare examples of action research is the series of studies around the HISP project (Braa and Hedberg, 2002).

2.2.6 IS success

Success and failure is a familiar theme in IS, and has also been addressed in the context of developing countries (Avgerou, 2008). Sustainability is one relevant aspect of overall IS success in this respect (Heeks, 2002b; Krishna and Walsham, 2005). Sustainability is closely related to the scaling of pilot projects to other contexts (Braa et al., 2007; Walsham and Sahay, 2006). In the following, the relationships between success, sustainability and scalability will be outlined in the realm of IS, particularly in the context of developing countries.

Heeks (2002b) defines success for an IS initiative in developing countries as one in which "most stakeholder groups attain their major goals and do not experience significant undesirable outcomes" (p. 102). Two key problems have been observed in successful implementation of IS (Sahay and Avgerou, 2002):

1. Many organizations have difficulties in nurturing and cultivating complex technology projects over the long periods of time that are typically required;
2. Resulting systems have little impact on the organizational weaknesses they were intended to alleviate.

The IS field has understood that there are several dimensions to success. For example, Bostrom and Heinen (1977) distinguished between social and technical issues, by viewing IS as socio-technical systems. In the developing country context, Barrett, Sahay, and Walsham (2001) have identified trust relationships as being important, both between stakeholders and towards new technology. Political and cultural issues have also been documented (Nicholson and Sahay, 2001).

There are two different approaches towards measuring success: First, normative models that seek to establish relationships between independent and dependent variables (DeLone and McLean, 1992). Normative models have been criticized for being too prescriptive and failing to take account of differing contexts. Second, contingency models try to avoid a single blueprint for success and change, so that situation-specific factors can be included in finding strategies for success. An example of a contingency based model is ITPOSMO (Heeks, 2002b), which declares several dimensions of gaps between reality and the design of a newly implemented information system. Larger design-reality gaps weaken the probability of success. When transferring industrialized country design to developing country realities, in many cases there are large gaps.

According to Krishna and Walsham (2005), factors that contribute to the success of IS initiatives in developing countries are committed leadership, the involvement of multiple groups, and a people orientation that tries to work with existing skills in the given context. Bass (2009) argues that in cases where new skills have to be built, they should be built incrementally over time. If senior management support is given and the project coincides with the technical objectives of team members, then skills and confidence grow and the level of technical supervision declines. Joubert (2007) criticizes an excessive focus on social factors in the literature on IS success in developing countries and emphasizes "certain basic technical requirements that are non-negotiable". He recommends balancing two dimensions of capabilities, a technical skill set, and project management skills. Bridges.org (n.d.) recommends a set of 12 best practices, or *habits*, to accomplish highly effective ICT-enabled development initiatives, covering all phases of such interventions.

2.2.7 IS sustainability

Sustainability is different from success. Sustainability cannot be investigated without its relation to overall success. For example, it is possible to sustain a project that does not deliver goals for most stakeholders – in other words, to sustain a largely unsuccessful project. In many cases, ineffective IS have been institutionalized. But only if an IS is useful, it is desirable to sustain it. Serving an actual organizational need or opportunity should go hand in hand with IS sustainability (Kimaro and Nhampossa, 2005).

In contrast, failure to sustain a completed project, e. g. after installation, is a form of sustainability failure (Heeks, 2002b). Hence, a completed project is a weaker measure of success than a sustainable project (Bass, 2009). For the developing country context, Krishna and Walsham (2005) suggest two criteria for successful IS implementation. First, most stakeholder groups shall attain their major goals and do not experience undesirable outcomes. The second criteria is the sustainability of the IS initiative. Sustainability is thus important for overall IS success, especially in developing countries. Walsham and Sahay (2006) consider sustainability an important, but neglected key issue. Only sustainable ICT projects can support long-term socio-economic development. Sustainability is a particularly critical issue for developing countries, given the strong reliance on temporary donor funds. The external support is eventually withdrawn, which poses a risk for the continuity of projects (Heeks, 2005b). With the dominance of donor funded projects, some describe

sustainability as "the ability of a project or intervention to continue in existence after the implementing agency has departed" (Harris, Kumar, and Balaji, 2003, p. 2).

Action research has been identified as particularly relevant for ICT4D projects (Walsham and Sahay, 2006). However, action research ICT4D projects also suffer from sustainability problems. Braa, Monteiro, and Sahay (2004) have identified sustainability of interventions as a vital, yet underdeveloped, quality criterion for action research in IS. Too often, projects only have impact as long as the action researchers' attention remains. Another frequent problem is that only prototypes are implemented instead of information systems that are properly institutionalized and used as a matter of routine.

The issue of sustainability has been recognized as relevant by many organizations in the field of development cooperation, not only for, but including, ICT projects. For example, the German Ministry for Economic Cooperation and Development (BMZ) states that "[i]n order for projects to have an optimum and sustainable impact, the BMZ places special emphasis on capacity building, educational measures in the national language and empowering local partners" (BMZ, n.d.).

There are broadly two different approaches to explain IS sustainability: factor models and process models. An example of the former is given by Kahen (1995) who provides a set of six dominating factors for the process of IT transfer: (1) Economic (e. g. availability of financial capital); (2) Manpower (e. g. availability of trained personnel); (3) Physio-ecological (e. g. availability of resources); (4) Cultural, Demographic, and Social (e. g. openness towards technology); (5) Political (e. g. government stability); (6) Existing information infrastructure (e. g. availability of telecommunications networks). For ICT projects in developing country contexts a varying set of sustainability categories has been put forward (Bailur, 2006; Batchelor and Norrish, 2003; Cisler, 2002; Kumar, 2004):

- Financial sustainability.
- Social/cultural sustainability: achieved when social exclusion is minimized and social equity maximized.
- Political/institutional sustainability: achieved when prevailing structures and processes have the capacity to continue to perform their functions over the long term; related to the phenomenon of resistance to change, particularly if vested interests are at stake.
- Technical sustainability.

A similar set of factors is suggested by Agha (1992). But an additional focus is

put on the usefulness of the information and an orientation towards user needs. Therefore, in Agha's analysis, utility is a key factor to achieve sustainability.

Whereas an innovation is first adopted and diffused partly for its technical merits and partly due to powerful actors, subsequently institutionalization plays an important role. Institutionalization is the process through which a social order or pattern becomes a social *fact*. Through institutionalization an innovation is adopted and maintained because of its acquired legitimacy, without having to rely continuously on strong personalities (Avgerou, 2000b). Institutionalizing routines of use is thus a key process that contributes to make information systems sustainable (Kimaro and Nhampossa, 2004). Heeks (2005b) summarizes the above discussion on factors in the form of three conditions for IS sustainability, which are easier to operationalize than the categorization in financial, social/cultural, political/institutional and technical sustainability categories:

Capacity: the IS project needs ongoing availability of the required resources, such as money, skills, data and technology. This makes the IS usable.

Utility: the project must keep meeting the needs of at least some stakeholders. This makes the IS used.

Embedding: for long-term sustainability, the IS must be institutionalized, i. e. embedded in the rules, norms and culture of the local setting. This makes the IS routinely used.

Information systems that have been institutionalized in organizations can be considered in some ways as *rigid*, as having barriers to change. Changing an installed IS is restricted by technical and institutional conditions. This can prohibit the necessary change that is needed for the IS evolution. Lack of change is a threat to sustainability (Nhampossa, 2005). This underscores the dynamic nature of IS projects and the relevance of a process orientation in order to achieve success and sustainability.

An approach to process based models is to consider the fit between design and reality of factors. Here, a process of creating this fit underlies IS sustainability. Such a process typically includes two kinds of adaptations: Adapting reality to the design, for example through training of users and other participants in order to provide the required skills. The other adaptation process is to change the design to the particularities of the reality, such as providing user interfaces in local languages (Heeks, 2002b).

Adaptation processes are sometimes being viewed as neatly delineated into independent stages of (a) design adaptations on behalf of the IS developers during the early phases of an IS project, and (b) adaptations of local reality during later project phases, par-

ticularly as part of organizational implementation. An approach based on this view starts at a supposedly stable situation, prepares the change, implements the change in order to return to another stable, albeit improved, situation. But this view has been questioned in favor of seeing design and implementation activities as interlinked; Organizations need to gain experience with IS in order to better understand how to use them best in practice. This is because most IT based tools are open ended in nature, allowing them to be customized to different organizational contexts. A model for change shall therefore accommodate, or even better, encourage ongoing and iterative experimentation, use and learning (Orlikowski and Hofman, 1997). Therefore, the process of achieving sustainability by creating fit between design and reality cannot be limited to certain phases of IS projects, but needs to include a wide spectrum of the IS life cycle. Another relevant type of process is *appropriation*, i. e. taking ownership of the IS project by local stakeholders, and undertaking their own adaptations. Appropriation improves the probability of IS sustainability (Heeks, 2005b).

An example of a process oriented approach to sustainability within IS action research is provided by Braa, Monteiro, and Sahay (2004). The approach, called "networks of action", considers scalability not as a 'luxury', but as a precondition for sustainability. To achieve scalability, the networks-of-action approach emphasizes a multiplicity of interconnected sites and opposes single site prototype cases (see also section 3.6.2, p. 95). Furthermore, the approach underlines local appropriation through exchange of knowledge between sites and adaptations to the local context.

2.3 Development and ICT/IS

ICTs have a strong presence in approaches to development. ICTs are involved in how individuals carry out their work and leisure activities, in the way people organize themselves in groups, and in the way organizations and societies are formed. The term development has been used in a variety of different ways. For many, development is primarily associated with economic development (Walsham, 2005). In the context of developing countries the term is often used in a wider sense, describing development loosely as the ideal of improving people's situations (Sutinen and Tedre, 2010). Not least is the term development also essential in the technical sense, like in software development. Therefore, throughout this thesis, development is used to either refer to more focused technical development or more broadly to the improvement of people's situations.

In many countries, particularly in the more industrialized countries, technologies have historically been strongly associated with development. ICTs make no exception, and they are also seen by many as important contributors to development in the poorer countries of the world. This is expressed by former United Nations general secretary Kofi Annan as follows:

> "The new technologies that are changing our world are not a panacea or a magic bullet. But they are, without doubt, enormously powerful tools for development. They create jobs. They are transforming education, health care, commerce, politics and more. They can help in the delivery of humanitarian assistance and even contribute to peace and security." (BMZ, n.d.)

International donors and financial institutions have a tendency towards an approach to development in poorer countries which attempts to achieve development through economic growth, enabled by the creation of liberal democratic systems. This approach incorporates the belief that development is about the elimination of poverty, particularly the so-called *absolute poverty*. The United Nations' Millennium Development Goals (MDGs) are specifically concerned with poverty reduction. (Unwin, 2009, p 14). ICTs are reflected in the MDGs in goal 8, *Develop a Global Partnership for Development*, more specifically in the sub-goal 8.F: "In cooperation with the private sector, make available benefits of new technologies, especially information and communications". This is however only one aspect of how ICTs can potentially contribute to development. Whereas for many development practitioners, development has become synonymous with poverty elimination through good governance and economic growth, there exist "a plethora of alternative arguments that see development as something much more subtle, culturally shaped and socially relevant" (Unwin, 2009, p. 15). ICTs are also relevant in such alternative approaches. For example, the MDGs lack an emphasis on local participation and empowerment, excluding women's empowerment (Deneulin and Shahani, 2009). Empowerment is however a topic that can hardly be overlooked in the development of those that have little say in the contemporary world. ICT based interventions are often challenged to be designed in a way that empowers communities or organizations towards a greater potential to overcome current problems in a sustainable manner. The following quote from Saraswati (1997) illustrates the sentiments of disadvantaged communities and the relevance of empowerment:

> "For our guidance, then, let us make use of Gandhi's concept of swaraj: To get Swaraj then is to get rid of our helplessness. The problem is no doubt

> stupendous even as it is for the fabled lion who having been brought up in the company of goats found it impossible to feel that he was a lion'. Such is the state of the ancient cultures with great traditions which are now condemned as 'developing countries'."

2.3.1 Clarification of terms: ICT4D and its relatives

Multidisciplinarity is the backbone of development informatics according to Johanson (2010). As with other multidisciplinary fields, the boundaries of the development informatics field have remained yet unclear (Toyama and Dias, 2008). One of the effects of this uncertainty has been the emergence of a variety of different designations for development informatics research. The term *development informatics* (DI) is only one of several names with broadly similar meaning. A popular name is *ICT4D*; other names include *ICTD*, *ISDC* and *e-development*. All of these acronyms describe variations of the theme of using ICTs in developing country contexts. ICTs thereby refer to the following research foundations (Heeks, 2006):

I: Work drawing from library and information sciences

C: Work drawing from communication studies

T: Work drawing from information systems

Information and Communication Technology and Development (ICTD) relates to a wide array of ICT applications in the context of developing countries. It leaves open whether the applications aim at development goals or not (Coward, 2009; Unwin, 2009). Development Informatics (DI) has a similarly broad meaning of ICTs in the context of socio-economic development. Here, the signification of *informatics* itself may need some explication. The word *informatics* has been coined by German and French scientists several decades ago, and later taken over into English (Widrow, Hartenstein, and Hecht-Nielsen, 2005). An English definition of the field of informatics is "the study and application of information technology to the arts, science and professions, and to its use in organizations and society at large" (*Informatics defined*). Informatics uses computing to solve problems in areas such as security, health care, education, poverty, and challenges in the environment. Informatics differs from computer science due to its strong focus on the human use of computing.

More narrowly focused, the often used term Information and Communication Technology for Development (ICT4D) refers to the exploitation of digital technologies to de-

liver development goals, often more specifically oriented towards the MDGs. The term e-development is used rather seldom. One example of its use occurs in Heeks (2002a), where it is defined as the "use of electronic ICTs like the Internet to support development" (p. 1). This suggests a similar intention as ICT4D, which explicitly concerns itself with the contribution to development goals. Information Systems in Developing Countries (ISDC) is an acronym used by Avgerou (2008). It refers broadly to the application of ICT based information systems in the context of developing countries.

For this thesis, the term development informatics corresponds well with the practical work that is the focus of this research. Furthermore, in comparison to the term ICT4D, it is "less technocentric and allows an equal focus on information, knowledge, and information systems as well as on ICTs" (Heeks, 2006, p. 2). But other acronyms are also mentioned, particularly when other sources are referenced. Especially the term ICT4D is popular and is thus being mentioned frequently here.

2.3.2 The multidisciplinary nature of development informatics

"The field of ICT4D is inherently multidisciplinary" (Unwin, 2009, p. 3).

ICT4D intends to contribute to development by utilizing the power of ICTs. Therefore, there is an implicit need to address not only aspects concerning the ICTs being introduced into a particular context, but also about the development to which the ICTs supposedly contribute. This puts the ICT4D field at a crossroads of at least two – if not more – disciplines. There is a debate about the range of disciplines that shall be covered within ICT4D, which is summarized in the following.

Parmar (2009) criticizes single disciplinary ICT4D projects and argues for a particular multidisciplinary approach to ICT design and deployment. He proposes a design framework based on four scientific disciplines: computer science, social science, industrial design and marketing. Thereby, social science's role is to contribute to contextual understanding at the individual and the community level. Computer science's role is to enable the development of hard- and software. Industrial design plays a part in user interface design and usability testing. Finally, marketing is supposed to act as an enabler of economic sustainability and technology adoption.

Heeks (2008) suggests that ICT4D innovators should be *tribrids* who have sufficient knowledge about three areas: (a) computer science, (b) information systems and

(c) developing studies, in order to tackle three important questions:
Computer science: What is possible with digital technology?
IS: What is feasible with digital technology?
Development studies: What is desirable with digital technology?

First, computer science is essential, covering the spectrum from hardware and firmware to software and human computer interaction. But technocentricity easily leads to systems that may technically work well, but lack a development contribution. Secondly, the information systems perspective provides models to understand human, political and contextual factors of why many projects fail, and provides certain approaches how to prevent such failures. However, the field of information systems not only largely disregards the technical artifact, but also shows little interest in processes of development: "IS tends neither to understand, nor to use the ideas of development studies" (p. 31). Thirdly, the development perspective increases the relevance of ICT4D endeavors. Although development studies have tended to be skeptical about technology, technology has recently been regarded more seriously as an enabler to achieve development, e. g. by advocates such as Jeffrey Sachs who considers technology as central to achieving the MDGs. It is unclear how such tribrids could emerge in a planned way. Education is one important contributing factor, but tribrids also "tend to self-create during ICT4D projects as leaders from any individual domain rapidly find themselves facing problems that only insights from other domains can solve" (Heeks, 2008, p. 31). Heeks calls for such tribrids not only on the project level, but also on the policy and program levels.

The aspect of communication has also been emphasized for developing cooperation projects. Although big international organizations refer to the technology's potential for development, it remains important to consider how technologies are introduced. Many ICT projects have had limited success in achieving intended goals towards bridging the digital divide. Dropping technology into rural contexts doesn't lead to sustainable solutions. Neither does the provision of access to information automatically generate change. Therefore, information technology projects shall be combined with communication, which implies participation, sharing of knowledge in a horizontal way, and respect for diversity and culture. In this way, communication can be a facilitator for ICT4D projects, to better contribute to sustainable development and to lower different aspects of the digital divide, not only the technological divide, but also the social and cultural divide (Gumucio-Dagron, 2003).

Figure 2.2 shows an approximate map of foundational disciplines for development

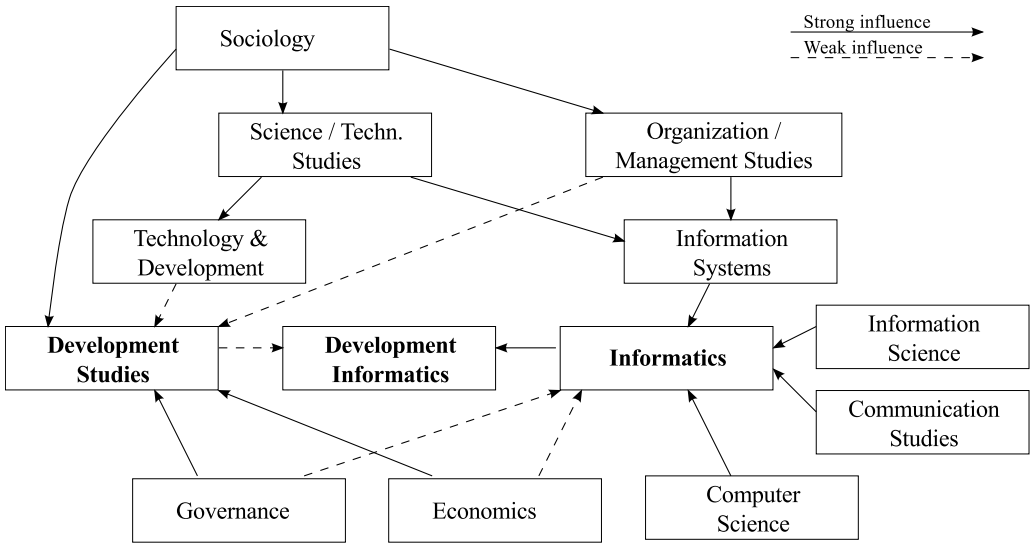

Figure 2.2: Disciplinary foundations for development informatics research (Heeks, 2010).

informatics (Heeks, 2010), which is not necessarily complete, since other disciplines and sub-disciplines could be added. There has yet been a rather weak connection from development studies to development informatics. Studies have rather looked at readiness, availability and recently also on uptake of ICTs than at development impact.

The intellectual community has been observed to be divided between social scientists and engineers, with little integration of information systems and computer science within the informatics field; social scientists often have reservations or difficulties addressing technological aspects or even doubt the mere necessity of technological innovations for ICT4D work, whereas engineers often felt that their work has been less favored. These single-discipline centric views have been criticized as worrisome (Best, 2010; Schunter, 2007). Although multidisciplinary work may seem to be an obvious choice for ICT4D, there are fundamental difficulties to overcome, for example publishing issues in multidisciplinary research teams (Walsham, 2010). However, drawing on theory from multiple disciplines and interacting at conferences are further options for multidisciplinary work (Best, 2010).

Despite the different opinions on which research disciplines are relevant for the ICT4D field, the prominence of the development context suggests that development aspects should receive attention in DI studies. This includes an elaboration on the kind of development that ICTs aim to contribute (Walsham and Sahay, 2006). Furthermore, as Sutinen and Tedre (2010) point out, the character of ICT projects in developing countries is relevant.

Table 2.2: Division of ICT4D approaches (Sutinen and Tedre, 2010)

	ICT: Existing	ICT: Not Existing
Development challenge: Unknown	1) Matching a tool and a problem	3) Exploratory research
Development challenge: Known	2) Evaluation research	4) Constructive research

ICT4D projects may or may not include the construction of technical artifacts. Therefore the focus can vary between construction and evaluation. Many projects will have both elements to a certain extent.

2.3.3 Computer science and development

Within the computer science field, ICT4D has yet attracted only limited interest and occupies a small niche. Computing and computer science research is often construction oriented, building computational tools through human activity, which in turn are agents for social change (Tedre, 2009). This construction orientation is complementary to evaluation and exploration studies. Because of the limited involvement of computer scientists, ICT4D research has engaged less in designing new technologies than in evaluating existing ones in development contexts. The reason for the yet low interest on the part of computer scientists remains only a speculation, but may be due to the perceived absence of computationally challenging tasks to be solved. The computer science perspective is nevertheless relevant because it "focuses on exploring the resources, or inputs, of a particular context and on basing the design of a technical intervention on the available resources, so that the output makes a difference in the development context" (Sutinen and Tedre, 2010, p. 221).

Table 2.2 illustrates the relationship between computer science and development studies. It is based on two dimensions: The first dimension is about whether the development challenge at hand is well-known or not. This is combined with the second dimension, whether technical solutions exist for the problem or not. The combination of the two dimensions yields four different types of development informatics research:

1. Matching a tool with a problem occurs in two flavors: Either the goal is to understand social or economic needs and finding an existing technological solution. Or a

technological innovation has been designed, and the goal is to find a problem, which can be solved by the innovation.

2. If both the developmental challenge and the technical solutions are largely known, evaluative research aims to get a better understanding of the changes that go along with the introduction of the technical tools. The outcome of evaluative studies is often qualitative or quantitative reports.
3. Exploratory research is often necessary as groundwork for other types of research. An important aspect is to understand the problem space and delimit problem boundaries.
4. In case the need or problem is well understood, but no technical solution exists yet, constructive research aims to define, design, implement and test tools to tackle the issue at hand. This is a core computer science activity.

The four types in table 2.2 are ideal types, and many studies incorporate more than a single type. For example, a research study may construct a technical artifact and subsequently evaluate it. Another common scenario is exploratory research followed by constructive research. Exploratory investigation is needed because much of the research in computer science is not of the sort *seek the best solution to a previously specified problem* (Fletcher, 1995). In many cases computer scientists work with problems that are poorly understood, so that it is mandatory to first understand and delimit the problem more precisely. Another conclusion from the division in table 2.2 is that not all development challenges require technical innovations, but often existing technologies are sufficiently well suited, which may be adapted as required.

Without the capabilities of computer science, ICT4D research remains technology driven. This is because without the possibility to design new technical innovations, researchers are limited to the technological tools at hand. Without a good understanding of possibilities and limitations of ICT, technical equipment is often seen as a constant, unchangeable given. With such a limited view there is the danger of a *one size fits all* approach and technological dependency in ICT4D research. A technical or constructive orientation towards development informatics does not need to follow a technocratic orientation where the project team follows a predefined technical plan. On the contrary, computer science helps to avoid too much techno-enthusiasm (Sutinen and Tedre, 2010).

Computer science aims to "come up with functional technology that contributes to changing conditions within a given context" (Sutinen and Tedre, 2010, p. 226). Thereby, the attribute *functional* promotes both access and ownership. Access is related with usefulness

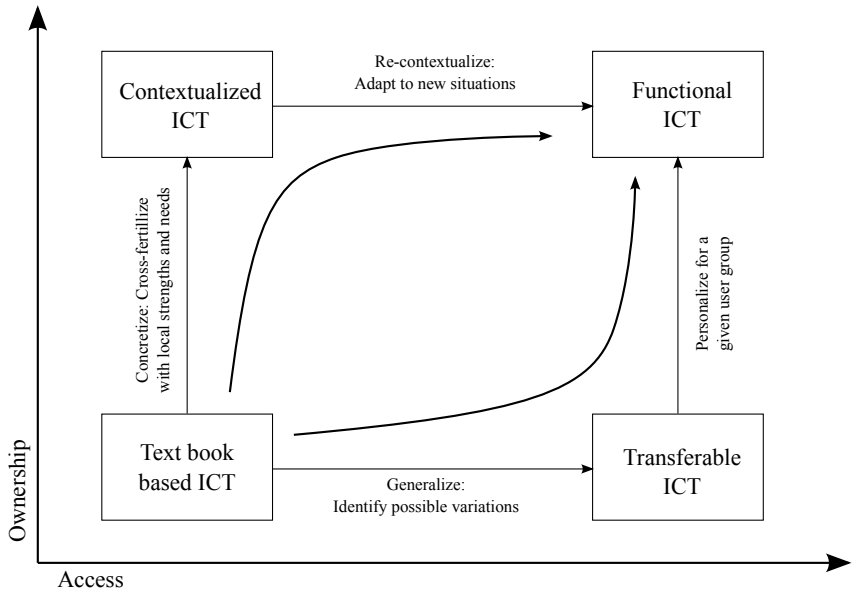

Figure 2.3: Two alternative paths towards a functional ICT artifact (Sutinen and Tedre, 2010)

and usability, whereas ownership is more context-dependent and has less objective criteria. There are two different ways towards functional technology (see figure 2.3):

Generalize-first: An existing technical solution is generalized to provide maximum access, and only afterwards personalized or adapted to particular user groups. The danger of this approach is that the solution may be unable to accommodate local particularities.

Concretize-first: The priority lies in providing a working solution for a given context, based on the available inputs. Thereby, ownership is strengthened early in this context. Later the solution is generalized to work in different contexts. This approach is particularly useful for studies with a strong exploratory aspect.

Steps along the *access* axis emphasize the product-oriented, deductive, top-down aspects of the process. Steps along the *ownership* axis represent creative, inductive, bottom-up perspectives.

2.3.4 Innovation and Appropriate Technology

There are two extremes along the continuum of approaches to technology and development: The passive diffusion approach and the active innovation approach. The passive diffusion view is exemplified by the spread of mobile telephony, where solutions

such as mobile telephone technology are designed elsewhere and imported and diffused into a developing country context. Active innovation is related to local intervention and occurs if the market doesn't deliver satisfactorily. Most ICT4D innovation is likely to happen in applying or adapting existing technologies, rather than within large scale hardware and operating system ventures such as the One Laptop Per Child (OLPC) project, which have proved to be risky.

Three different variations can be distinguished in the way that innovation takes place: *pro-poor*, *para-poor* and *per-poor* innovation. Pro-poor innovation occurs outside poor communities, and represents a supply-driven focus. Early telecenter projects were designed in this way. The advantage is that outside expertise can be engaged easily with this approach. But there is a danger of gaps between design and reality, more specifically between designers' assumptions and on-the-ground realities. A successful example of pro-poor innovation is given by the pricing models of prepaid mobile phones, which have facilitated the uptake of mobile telephony in the developing world (Heeks, 2008).

Para-poor innovation occurs alongside poor communities. A key lesson from early ICT4D efforts has been the need for participative design processes. Rather than designing what is supposedly best for the recipient community, future users need to engage themselves in the design process. Participation is thereby both a means and an end of development (Moens et al., 2010). Effective participation is however more easily said than done. Participation creates divides between designers and users, such as rich versus poor and Western versus non-Western mindsets. Common points to consider include:

- Who participates: often it is a small minority elite;
- How do they participate: individually or in group processes;
- Why do they participate: participants often provide reasons just to please donors

Per-poor innovation is done by the local community or organization itself. This kind of innovation has only become possible in recent years, since new technologies have started to reach the poor. First, mobile phones, later PCs, and now the Internet are being more and more available to developing country citizens. These technologies are then used by local users to innovate. Per-poor efforts typically occur by adapting existing technologies and applying them in new ways. There are still few examples of per-poor innovation. One example is the use of prepaid mobile phone airtime as alternative currency, which makes new ways of transactions possible, such as long distance transactions between relatives.

The Appropriate Technology movement has in the past already successfully made

the transition from solutions for the poor (*pro-poor*) to solutions alongside the poor (*para-poor*) to solutions by the poor (*per-poor*). Within the field of DI, the methods of Appropriate Technology might be enhanced with ideas from Open Source and Web 2.0 innovation models (Heeks, 2008).

2.3.5 Community Informatics

Community Informatics (CI) is the application of ICTS to enable and empower community processes. According to Gurstein (2007), the objective of CI is to use ICT to enable the achievement of community objectives including overcoming *digital divides* both within and between communities. CI can be used to examine how and under what conditions ICT access can be made useful to the range of excluded populations and communities and particularly to support local economic development, social justice, and political empowerment using the Internet. CI is a framework for systematically approaching information systems from a community perspective, where the community is the *owner* or operative agent. This is an alternative to the traditional view that information systems are owned and operated by organizations.

McIver (2003) expresses the need for CI in contrast to Management Information Systems (MIS), which has established best practices that generally assume an abundance of resources and expertise to which communities often do not have access. According to McIver the *grand challenge* in CI is to develop technological solutions for communities that are economically, socially, and culturally appropriate and that are operationally and economically sustainable. This is especially true for developing countries, where resources and training may be even scarcer than in most communities.

Conventional approaches to ICT4D tend to be dominated by a western, donor community set of values and priorities. ICT4D policies often follow a top-down philosophy that starts by defining national policy plans, followed by creating enabling conditions in the market, and finally creating projects that follow policy guidelines. This macro-level oriented ICT4D strategy does not necessarily give access within the information society to individuals and groups at the micro level, and may thus prevent development opportunities in developing countries that could be possible with more inclusive bottom-up approaches. This top-down, technocratic ICT development discourse has been emphasized by organizations such as the World Bank. It has received critics, for example by M. Thompson (2004),

who outlined that this approach excludes alternative views of technology and development. Vaughan (2006) positions CI as an alternative by referring to best practices and lessons learned from a plethora of case studies in the ICT4D field, even though these practices are sometimes not explicitly called CI methods.

Community Informatics is an alternative to the top-down approach. CI attempts to embed ICT in existing community structures, utilizing existing social capital in those structures. Rather than imposing externally designed ICT solutions, ICT is introduced with the objective to help the community identify and meet its needs and to target effective use. Gurstein (2007) contrasts the CI approach to ICT with the design and deployment of information systems in industrial contexts, where the difference is that CI emphasizes bottom-up processes for system design, whereas in industrial settings system design is guided by corporate management.

2.4 IS development from a computer science perspective

In addition to particular technologies that have been reviewed in the developing country context (section 2.2.4), further technological details shall be reviewed that are relevant for sustainable IS development. As information systems are both an organizational and a technical matter, the technical creation of information system artifacts also needs to be sustainable over extended time frames. One of the early problems encountered in software development was the exponential cost for the maintenance of software applications – which motivated the structuring of the development process into a variety of software life cycle models. Until today, there is no *silver bullet* that would make it possible to approach software development exclusively as an engineering activity; unlike many physical artifacts, software in most cases is ever-evolving along emerging requirements over time, possibly with profound changes to the internal technical architecture. Therefore, software development methodologies are needed that can sustain software system development in a reasonable pace. In this section, lessons learned in software development practices are reviewed. Furthermore, the emergence of iterative and incremental development (IID) and agile methods is reviewed. Agile methods are also put into perspective of global software development and sustainable software development. At the end of the section, open source is considered from the particular perspective of the higher education sector.

2.4.1 Historical overview of software development methodologies

Historically, the evolution of software development methodologies can be distinguished in a pre-methodology era in the 1960s and 1970s, the early methodology era during the late 1970s and early 1980s, the methodology era until the mid 1990s and the post-methodology era afterwards (Avison and Fitzgerald, 2003). The *pre-methodology era* was characterized by an absence of formalized or explicit methodologies, for example requirements were rarely well defined. The emphasis was on solving technical problems. The drawback of projects in this era was poor control and management of projects. In the *early methodology era* different stages of building computer applications were identified. The systems development life cycle (SDLC) emerged, often referred to as the waterfall model. The phases typically consisted of (Royce, 1970): Definition or feasibility study; analysis and requirements specification; design; construction; testing; installation; operation and maintenance. The phases were associated with particular outcomes. For example, the requirements analysis document was the expected outcome of the analysis phase, which served as input for the subsequent phases. Each phase has to be completed, before the next could be entered. There water fall model can be considered document driven (Larman and Basili, 2003). Goals of the SDLC are (Brandon, 2006):

- Do it right the first time
- Meet the stated requirements
- Timely completion
- Complete within cost constraints
- Build systems with the necessary quality

The waterfall model has major problems, amongst them the frequent failure to meet the real needs of the users, inflexibility because of the difficulty and the cost of changing the design, and application backlog due to the mounting workload for maintenance (Avison and Fitzgerald, 2003). However, the waterfall model continues to be prominent, and many variations exist of the basic waterfall method. The original formulation of the waterfall model already proposed to run twice through the complete set of stages (Royce, 1970). The outcome of the first run serves as a prototype for a second, improved version. Reasons for the widespread adoption of the waterfall model, despite evidence against its usefulness, include the following (Larman and Basili, 2003):

- It's simple: First do the requirement, then design, then implement

- It gives the illusion of an orderly process
- It was widely promoted as an ideal model

In the *methodology era*, methodologies got diversified, with the overarching objectives to better meet user needs, to improve the software development process in terms of developer control and productivity, and to standardize various software development processes within organizations. The wide variety of methodologies that emerged in the methodology era can be classified in seven broad themes, which are not necessarily mutually exclusive (Avison and Fitzgerald, 2003):

Structured: Structured programming concepts were applied to analysis and design, and new techniques such as data flow diagramming were introduced to improve top-down project management.

Data-oriented: Focus on data as the key element for system development and the entity-relationship modeling as the primary technique.

Prototyping: Early building of an approximation to allow users to react prior to the system's realization.

Object oriented: Identification of objects, attributes and classes, with the aim of improving reuse.

Participative: Focus on the involvement of users and other stakeholders.

Strategic: Align software applications with an information systems strategy that supports overall organizational objectives, e. g. by applying business process reengineering.

Systems: Adopt a holistic view beyond the single-application boundaries to integrate the complexity of human activity systems.

The *post-methodology era* brought a reappraisal of development methodologies by researchers and practitioners. There is wide diversity from total neglect of methodology to new forms of methodology, moving away from bureaucratic forms of the methodology era. Some, especially programmers, have rejected methodologies, often because following methodologies resulted in low productivity. Methodologies often lacked contingency, being inappropriate to solve given issues in a particular problem situation. Few methodologies address critically important social, political and organizational dimensions of development. Methodologies have in many cases turned out to be too inflexible to allow changing requirements. Most methodologies make assumptions, for example a stable environment or certain user knowledge about their own requirements. The problem is that the assumptions are invalid for a lot of circumstances. One particular area where methodologies are not

widely used are web development projects. Here, ad-hoc, trial-and-error approaches are predominant, similar to the pre-methodology era.

Alternative, emerging methodologies include an improved object oriented approach with a focus on reusable components, incremental development that is more conducive to changing requirements, and contingency that allows for different approaches depending on particular problem situations. Methodologies remain influential, new methodologies are being designed, and the current diversity makes it more likely to identify an appropriate methodology for a given problem situation (Avison and Fitzgerald, 2003).

2.4.2 Iterative and incremental development

Evidence has been produced against strictly following a sequential life cycle model, which starts with a supposedly precise statement of requirements and the subsequent design and construction of the product. It has been argued that it is utterly difficult to succeed in designing a real product in this way, and the closest approximation to the waterfall model is to 'fake' the process in a way to write some of the requirements specifications after the designated requirements phase (Parnas and Clements, 1986). Statistical data suggests that more flexible approaches are associated with better performing projects. Flexibility in this respect refers to the ability to generate and respond to new information for a longer proportion of the development cycle, unlike the concentration of requirements definition into an early stage in the waterfall model (MacCormack, Verganti, and Jansiti, 2001). There is also a sense that the sequential life cycle model, that is, a model without iterations, exceeds our human intellectual capabilities for management and control (Mills, 1976).

Iterative development is sometimes used to describe merely the act of revisiting work, but in its modern sense it means evolutionary advancement. Modern iterative and incremental development (IID) can be characterized as "feedback-driven refinement with customer involvement and clearly delineated iterations" (Larman and Basili, 2003, p. 49). In contrast to the *sequential life cycle* of the waterfall model, IID represents an *iterative life cycle*.

Iterative and incremental development has a long history, although it has only become more prominent recently in the post-methodology era. There have been large projects conducted successfully already in the 1970s and 1980s (Larman and Basili, 2003). In an early description of iterative development the process starts with a first, executable

model, which is tested and then further expanded through a sequence of models, until the model becomes the final system (Zurcher and Randell, 1968). It is worth noting that the paper that is considered the source of the waterfall model (Royce, 1970) did not propose a single-pass waterfall. It suggested to do it twice, which allows feedback and adaptation from the first pass to be integrated into the iteration of the second pass.

Swartout and Balzer (1982) presented case study results that support the argument that specification and design cannot be neatly separated into different phases, but are inevitably intertwined. Specification can change at least in two circumstances; first, design constraints may force a specification change. Second, design may provide unexpected, desirable opportunities, motivating augmentations to the specification.

Gilb (1985) promoted an *evolutionary delivery* model based on an *eternal development cycle* of delivering software to real users, measuring the added value and adjusting the design and specification. He contrasts the model with the *phased and revolutionary* waterfall model. In the evolutionary delivery model, iterations are intended to take days or weeks, as opposed to the waterfall model in which delivery takes months or years. Adaptation is not seen as a threat but is *native*. Because of frequent delivery the investment risk is much lower, and it's possible to stop in the middle with a useful system. Gilb argues that with the evolutionary delivery model management regains full control, which was lost in the waterfall model, where management was put at the mercy of the developers.

Boehm (1988) formulated the *spiral model of software development and enhancement*. The spiral model is risk oriented in the sense that each iteration needs a risk assessment step. The outcome of the risk assessment then guides further activities in the iteration. In case of high risk related to user interface or performance, the spiral model suggests to follow an evolutionary development model to integrate early user feedback. If other risks such as program development risks dominate, the waterfall model is a possible choice. The spiral model thus claims to be a general model, and several other process models can be considered special cases or the spiral model.

Curtis et al. (1987) viewed software development as a problem solving process involving multiple agents; developers try to solve problems presented by the requirements, but at the same time users try to solve a problem which they express in requirements. But users often do not have profound insight into the limits of technology and do not understand the subtleties of their problem. The authors conclude from observations of several large software development projects that such projects shall be treated as a learning and

communication process rather than a sequential manufacturing process. Successful projects emphasize cyclic learning and put high attention to people's skills and communication issues.

2.4.3 Agile software development

Agile methods are strongly linked to the history of IID methods. The appellations *agile* and *agile methods* were coined in 2001 by a group interested in modern, simple IID methods and who formed the Agile Alliance (`http://www.agilealliance.org`). Before that, IID methods that were later considered agile methods had already been developed – many promoters of the agile approach were involved in the evolution and use of various IID methods. Examples of methods with such a history are Extreme Programming (XP), Dynamic Systems Development Method (DSDM), Scrum and Feature-Driven Development (FDD) (Larman and Basili, 2003). Different variations of agile methods were independently developed in different continents: Dynamic Systems Development Method in Europe; Feature-Driven Development in Australia; and Extreme Programming, Crystal, Adaptive Software Development, and Scrum in the US (Williams and Cockburn, 2003).

In addition to the agile methods listed above, Abrahamsson et al. (2002) also includes open source software development in the set of agile methods, because it resembles various characteristics of agile methods. For example, like agile methods, open source development lacks a document-driven approach to software development, and from the outset open source projects make frequent releases. Warsta and Abrahamsson (2003) suggest that open source communities could benefit from the practical solutions put forward by agile methods and vice versa.

As an example of an agile method, Scrum was inspired by Japanese technology product development that had nothing to do with software development. It emphasized overlapping phases of technology development (Takeuchi and Nonaka, 1986). The Scrum method uses time boxed, fixed time intervals called *sprints* with a typical length between a week and a month. There is no predefined process in Scrum. Daily short meetings, which may also be time-boxed to around 15 minutes, lead the process. The goal of each sprint is to deliver as much quality software as possible (Beedle et al., 1999).

Extreme Programming (XP) received a lot of attention. It's guided by an emphasis on communication, simplicity and feedback. XP has a strong customer focus, and aims to

put the customer not only in control of the features, but also of scheduling and prioritization. With its practices XP intends to produce software at a sustainable pace (Beck, 2000). The set of practices includes pair programming, refactoring, unit testing, small releases, collective ownership, planning games involving the customer, and others. Small releases refer to small development cycles and frequent releases, which are made daily to monthly. Therefore, the release are typically done considerably more often than in other IID methods like the spiral model. Collective ownership encourages developers to touch and improve any code anywhere in the system if they see the opportunity. A further XP practice is that a customer representative sits with the development team full-time (Beck, 1999).

A common feature of agile methods is a focus on people's competency and on working jointly to improve their competency. This is expressed in the phrase *people trump process*; a process such as a software development method can be as useful as a framework for groups to work together, but a process cannot per se overcome lack of competency, whereas competency can overcome limitations of a process. Hence, agile methods do not recommend to rigorously adhere to process, but to focus on communication and skills (Cockburn and Highsmith, 2001).

The Agile Alliance published the so-called *Agile Manifesto*, which has been written by a group of 17 experienced software developers and subsequently signed by thousands on the dedicated web page (`http://www.agilemanifesto.org`). The agile manifesto has two main parts: its purpose and its underlying principles. The purpose is stated as follows (Fowler and Highsmith, 2001):

> "We are uncovering better ways of developing software by doing it and helping others do it. We value:
>
> - Individuals and interactions over processes and tools.
> - Working software over comprehensive documentation.
> - Customer collaboration over contract negotiation.
> - Responding to change over following a plan."

The purpose underlines that also seasoned and respected software developers do not have all the answers and hence do not subscribe to the silver-bullet theory. This is expressed by using the word *uncovering*. Secondly, the phrase *by doing* emphasizes that the authors of the agile manifesto practice agile methods themselves. Third, by using agile methods the members of the alliance intend to help, not tell others, and further the knowledge of all involved in the process. The 12 underlying principles of the agile manifesto

are as follows:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale
- Business people and developers must work together daily throughout the project
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation
- Working software is the primary measure of progress
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
- Continuous attention to technical excellence and good design enhances agility
- Simplicity – the art of maximizing the amount of work not done – is essential
- The best architectures, requirements, and designs emerge from self-organizing teams
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

2.4.4 Distributed software development

Software development is increasingly being conducted by distributed teams. Thereby, team members may be separated only by a floor or building, or may be as distant as being located on different continents with different time zones. In such scenarios communication is reduced. Particularly informal communication is missing, which often provides relevant information and allows to stay aware of what is going on in the project, what others are working on and what expertise others have. In addition to communication issues, other challenges of distributed software development include strategies to divide the work between different sites, cultural differences related to hierarchy, sense of time or communication styles, knowledge management such as the integration of customer feedback or outdated documentation, and technical issues like well-thought configuration and version manage-

ment of the software artifacts (Herbsleb and Moitra, 2001). Distance between colleagues increases complexity of organizational processes in three areas, coordination, control and communication (Carmel and Agarwal, 2001, p. 23):

> "*Coordination* is the act of integrating each task with each organizational unit, so the unit contributes to the overall objective [...]
>
> *Control* is the process of adhering to goals, policies, standards, or quality levels. Controls can be formal (such as budgets and explicit guide- lines) or informal (such as peer pressure). [...]
>
> *Communication* is a mediating factor affecting both coordination and control. It is the exchange of complete and unambiguous information – that is, the sender and receiver can reach a common understanding."

In addition to technological solutions such as a team web site or a distributed software configuration management system, Carmel and Agarwal (2001) suggest three tactics for addressing distance in distributed development: reduce intensive collaboration, reduce cultural distance, and reduce temporal distance. Intensive collaboration is due to the complexity of coordination. Temporal distance is related to working with partners in countries with small time zone differences.

In intensive collaboration scenarios, high levels of coordination are necessary. This happens when work is passed on between remote teams on a daily basis, for example when developers in America finish the day's work and pass on their work to colleagues in India and vice-versa. To minimize the coordination overhead, either certain maintenance units such as help desks and data centers can be located at the remote site, or the foreign entity can take over full responsibility, i. e. ownership of a system, product or process. The difference is that the workload of the remote team is relatively small with outsourced maintenance units, whereas ownership of systems with the foreign entity allows for high percentage of work performed by the remote team.

Cultural distance can be considered along two dimensions: organizational culture and national culture. One of the suggested arrangements to overcome cultural distance in distributed development settings is the so-called *bridgehead*. This reduces both organizational and national culture distance. The bridgehead is sometimes associated with a rule of thumb of 75 percent of personnel working offshore, and 25 percent of personnel working onshore at the customer's site. The onshore individuals are typically the more experienced and have a cultural sensibility for both sides. They act as a bridge and translate require-

Table 2.3: Impact of distance dimensions on distributed development processes (Ågerfalk et al., 2005)

	Temporal distance	*Geographical distance*	*Socio-cultural distance*
Communication	+ Improved record of communications	+ Potential for closer proximity to market and utilization of remote skilled workforces	+ Potential for stimulating innovation and sharing best practice
	– Reduced opportunities for synchronous communication	– Increased cost and logistics of holding face-to-face meetings	– Risk for misunderstandings
Coordination	+ Decreased coordination needs due to division of labor	+ Increase in size and skills of labor pool can offer more flexible coordination planning	+ Access to rich skill set and various practices
	– Increased coordination costs	– Reduced informal contact can lead to lack of task awareness	– Inconsistency in work practices can impinge on effective coordination, as can reduced cooperation through misunderstandings
Control	+ Opportunities for *round-the-clock* development	+ Communication channels often leave an audit trail	+ Access to rich skill set and authority
	– Management of project artifacts may be subject to delays	– Difficult to convey vision and strategy	– Different perceptions of authority/hierarchy can undermine morale

ments from the customer to offshore programmers. An important aspect is that face-to-face interaction at the customer site reduces miscommunication.

Communication, coordination and control activities are affected by several dimensions: temporal distance, geographic distance and socio-cultural distance. Temporal distance refers to a dislocation in time between individuals. It can be caused by time zone differences or time shifting work patterns. Geographical distance is a measure for the required effort of one actor to visit another actor. It is better measured by the ease of relocating rather than in kilometers. Socio-cultural distance is a measure of an actor's understanding of another actor's norms and values. It is possible for an actor A to be closer to actor B than B is to A. Table 2.3 shows a framework that summarizes opportunities and challenges in distributed development (Ågerfalk et al., 2005).

2.4.5 Agile methods in distributed software development

Agile methods may not seem to be particularly appropriate for globally distributed software development, since they rely on informal communication. Nevertheless, agile practices have the potential to improve long-distance collaboration. Layman et al. (2006) investigated a distributed team in the US and the Czech Republic that used Extreme Programming (XP). The team was able to establish close ties between developers and customer; face-to-face communication, which is a major point in XP, was approximated through email, a globally available project management tool, and an intermediary project manager who worked intensively with both groups. The case study proposes a set of recommendations for the use of communication-rich methodologies such as XP in distributed projects:

- Define a person who acts as the customer, who is able to make conclusive decisions on functionality, who is readily accessible, and who has a vested interest in the project. The importance of having such a kind of customer representative available is to keep high levels of relevance of the project and keep the customer in the driver's seat.
- 'Bridgehead' function of an experienced developer: A key member of one team physically located with the other team can improve communication between the groups. This role can act as an *on-site customer* for the developers, and as *on-site developer* for the customer's project management team.
- High process visibility, using globally available project management tools to provide the whole team with constant access to project information. For example, in the case study the team used a system called *XPlanner* to manage XP user stories, which were meaningful to both customer and developers.
- Short, asynchronous communication loops: Because not all communication can be accomplished in a synchronous manner through tools like Skype, asynchronous means such as email can be a sufficiently effective means to communication. Timely responses are a key condition to maintain steady progress and ensure feature relevance.

Holmström et al. (2006) investigated the application of agile practices at Intel and HP. Both companies had teams located in Ireland and the US that worked together on software development projects. The temporal distance due to time zone difference was a big challenge for communication and coordination. The main problem in this respect was the delay in responses. Geographical distance manifested itself mainly in the feeling of being two different teams. Establishing a trust and belonging (*teamness*) can be dif-

Table 2.4: Agile practices, benefits, and impacts on distance in distributed development (Holmström et al., 2006)

Agile practices	*Benefits*	*Distance*
XP pair programming	Help increase time overlap	Reduce temporal distance
Scrum simple planning	Help increase 'teamness'	Reduce geographical distance
XP pair programming and Scrum pre-game phase	Help increase mutual understanding and collaboration within and between teams	Reduce sociocultural distance

ficult. Socio-cultural distance was experienced mostly by language related issues, such as limited technical vocabulary and misunderstandings. It was seen that political and religious differences between individuals had the potential to exacerbate language-based issues and snowball to stronger tensions.

Several agile practices were used to deal with the identified distance-related issues. Some practices of XP were used for the technical aspects of software development, and parts of Scrum for project planning and tracking. The applied XP practices were pair programming, testing, refactoring, simple design, coding standards, and collective ownership. The other XP practices were not found suitable in these particular projects. Especially, the pair programming technique was satisfactory. For pair programming to work, some time shifting in their working hours was required by the participants due to the different time zones. It was found that pair programming resulted in high code quality, improved testing and debugging, and increased collective ownership. Pair programming was however not suitable for simple, well-understood problems. Refactoring was found especially beneficial in early phases of the project to avoid costly debugging in later stages. Daily Scrum meetings took place in a low-tech manner with post-it notes. Tasks were managed for a 24 hour period, and moved to the 'done' area when completed. The notes were managed on a shared web page, so that all participants could get a shared vision of the project's progress. Table 2.4 summarizes the benefits and impacts of agile practices on distance in distributed development.

2.4.6 Sustainable software development

Tate (2006) characterizes sustainable software development as a "mindset (principles) and an accompanying set of practices that enable a team to achieve and maintain an optimal development pace indefinitely", and considers sustainable development as important but unrecognized issue facing software organizations.

Tate traces reasons for unsustainable development to an excessive focus on feature development, a *code first then fix bugs later* mentality, too many dependencies between modules, a lack of safeguards such as automated tests, and supposedly temporary patches that are never addressed. Overall, unsustainable software development teams are primarily reactive to change, treating change as an afterthought. This faces the danger of a vicious cycle and a downward spiral demanding increasing manpower and decreasing productivity over time. In order for software development teams to become proactive about changes in their ecosystem, it is necessary to balance needs of the short and the long term; while features are developed, the cost of change within the software system is to be kept as low as possible, to lay the foundation for future changes and make it possible to quickly respond to changes in the ecosystem.

To achieve sustainable software development, Tate builds upon agile methods, by embracing change and applying agile practices, such as refactoring, unit testing, and small releases. Based on case study experience, sustainable projects have the following characteristics, which are strongly related to agile principles:

- Software works every day
- Heavy reliance on automated testing to catch problems while working on new features
- High standards of testing and code quality
- Avoiding over-design and building only what customers need
- Being uncompromising with defects, and making sure that all the known defects that are important to customers are fixed in a feature before moving on to the next one, preventing a mounting defect backlog
- Re-planning work as often as possible
- Always looking for ways to improve how the team and the software work

In summary, sustainability needs to be integrated in software development projects from the outset. Feature development and bug fixing shall go hand in hand and cannot be separated from underlying software health and infrastructure improvements.

2.4.7 Open source in the higher education sector

In section 2.2.4 (p. 45) open source has been characterized as a particular technology that is relevant to developing countries. In the following, open source is considered in relation to the specifics of the higher education sector.

UNDP has made the effort to analyze the possible role of free and open source software in the education sector, particularly as developing countries are concerned (Tong, 2004). The analysis identified opportunities for educational institutions in the areas of:

- ICT infrastructure, server software and desktop applications
- Administration of academic institutions, such as library management systems and learning management systems
- Teaching of information technology
- Open content, which applies similar principles as open source to the publication of content
- Research in free and open source software
- Training and certification in open source software, which gets its relevance due to the importance of building human resource capacity in open source software

Tong (2004) finds the following major points regarding the importance of free and open source software for education:

Lower cost: Open source software can lower the barriers to ICT access.

Reliability, performance and security: Higher quality is often achieved in open source software due to the open source development methodology, which facilitates the quick removal of bugs with the help of a large number of developers.

Build long-term capacity for the ever-growing significance of open source software in government, industry and other institutions. These organizations need qualified staff to manage OS applications.

Open philosophy: The approach of open source is similar to the open dissemination of knowledge in academia. Indeed the sum of human knowledge is the open sharing of ideas, theories and research.

Encourage innovations: Many innovations originate in universities, including well known free and open source software like Linux and the GNU operating system. The prevalence of OS will encourage students to tinker and experiment.

Alternative to illegal copying: If proprietary software is used, many students have no

other choice than using illegal copies of software applications.

Possibility of localization: Students in non-English speaking countries often have difficulties in using software applications due to the predominantly English user interfaces. Open source software offers better ways to localization.

Learning from source code: Having access to the source code gives students the opportunity to study high-quality source code of real life applications.

For administrative software in the higher education sector, a few proprietary vendors have dominated the relatively small market so far. This includes software for student information systems and library management. The cost is often very high due to the small market. Hence administrative systems are mostly out of reach for schools in developing countries, and even for some in developed countries. For library management and learning management, open source systems have already appeared and are available for use by academic institutions. This is not the case for student information systems, where no open source production quality systems were yet available (Tong, 2004).

A survey in Australian tertiary institutions (Glance, Kerr, and Reid, 2004) showed that open source software has already made significant inroads, with 94% of the respondents indicating that they were already using OS software. More specifically, all of the institutions had deployed OS server software, 50% had deployed OS software in the area of administration, 53% in teaching and 50% in research. Courant and Griffiths (2006) were tasked by a group of North American university leaders to investigate the potential of open source software in the context of higher education. A set of conclusions has been drawn from interviews with a variety of university members:

- Academic institutions have their own reality, which is not well met by commercial enterprises. Universities are overwhelmingly disappointed by the high cost and low flexibility of commercial solutions. This is in part due to the small size of the academic sector in comparison to other sectors in which ERP software is used. This corresponds to a market failure.
- The problems with current commercial software applications is to a large extent attributed to the distance between users and developers. This disjuncture between users and developers is common in many industries, but is especially grave in the academic environment. University collaborations can potentially lower this distance.
- Universities tend to limit their collaboration with other universities because they want to produce their own ‘stars’.

- Student information systems are possibly the strongest case for inter-university collaboration.
- The community development model (e. g. Linux, Apache) is less to address complex administrative functions such as payroll, human resources and student information systems. Directed, sponsored development is more appropriate for the initial development of such systems, because detailed, substantive engagement between supply and demand sides in development have proved to be important.
- The emergence of open source alternatives to commercial products may be able to correct the market failure.
- Support shall come from the commercial sector. It shall not be monopolized by a centralized unit. In fact, it is argued that not even initial support shall be given by a central unit because this will inhibit private enterprises from entering the market and offering services.
- A major question remains of how collaborative efforts are coordinated and controlled. Leadership by top level management of a considerable number of universities is considered to play a key role in order to establish the required infrastructure. Another important lesson is that any structure "must be commissioned and governed by an entity that has substantial authority, an effective governing structure and a clearly agreed upon sense of mission" (p. 6). However, it remained unclear how a coordinating body could look like.

Although community-driven software development projects in the higher education sector may be difficult to establish, they have been successful in cases where relatively small investments have been spread over many institutions (McDonald, 2009), for example in the cases of:

- uPortal (`http://www.jasig.org/uportal`): A web-portal framework sponsored by a consortium of educational institutions and commercial organizations.
- Moodle `http://moodle.org/`: A popular e-Learning platform, maintained by the core development team at the central 'Moodle Headquarters' and a worldwide network of partner organizations.
- Sakai `http://sakaiproject.org/`: An application suite comprising an e-Learning platform and other academic applications, supported by a foundation.
- Kuali `http://www.kuali.org/`: Administrative system suite, including financial systems, research administration and student administration, coordinated by the Kuali

foundation, which consists of both academic and commercial institutions.

In addition to potential financial gains, community-driven applications offer the possibilities of a better institutional understanding of the software and own improvements to the software that may not be available from commercial providers.

2.5 Summary and theoretical framework around sustainability

From the literature review the following conclusions are drawn:

1. To align IS project with the key issue of development, a common theme is to promote local knowledge, learning and adaptation, and not just providing physical capital and Internet connections.
2. For long-term success the entire system life cycle needs to be considered. Although sustainability may be primarily concerned with the time after design and implementation, sustainability is dependent on design and implementation activities, such as the level of utility and embeddedness that have been achieved during design and organizational implementation. Thus, IS sustainability needs to be based on the entire lifespan.
3. Sustainability is related to the ability to scale pilot projects to a larger set of installed systems.
4. Sustainability and scalability are affected by both the design process as well as the designed product.
5. Knowledge is neither static nor can it flow only in one direction, but has to constantly be shared between all actors involved in its creation and use.
6. Communication facilitates participation, sharing of knowledge in a horizontal way, and respect for diversity and culture.
7. Rather than getting merely access, local participants shall be empowered to make effective use of an information system.
8. A reflexive design process involving system design gradually through iterative processes is useful to successfully balance standardization and localization in global information system development.
9. An effective tool for agile IS development in globally distributed constellations is the

presence of a ‘bridgehead’. This reduces organizational and cultural difference between partners.

10. Organizationally oriented IS research as well as iterative and incremental software development methods emphasize that the particularities of the local context needs to be taken into account.
11. Open source is relevant for and has already made inroads into higher education institutions, especially in developing countries. However, not enough open source applications are available for the administration of academic institutions, particularly when it comes to student information systems.

Chapter 3

Research methodology

This section is concerned with aspects of the research methodology as they relate to the research project. At the beginning, observations are made about the context of the project being investigated. With this basis, possible research approaches are examined. The chapter is wrapped up with conclusions about an appropriate research methodology for the project.

3.1 Initial considerations concerning the OPUS project

The characteristics of the Mozambican OPUS project and the situation of the researcher are as follows:

Context: IS development occurs within the framework of a development cooperation project, with partners from diverse contexts.

Longitudinal: Period of several years.

Problem-solving component: Practical problems shall be addressed that arise during IS development, including organizational implementation.

Design perspective: The design and implementation of a student information system involves a combination of technology-based artifacts (system conceptualizations, technical capabilities, etc.), organization-based artifacts (structures, reporting, relationships, etc.) and people-based artifacts (training, consensus-building, etc.).

Levels of analysis: Inter-organizational collaboration and intra-organizational aspects are two important levels that deserve attention in order to be able to derive an overall picture of IS innovation. Furthermore, within organizations different groups can be

identified. Important groups include users, developers and managers.

These considerations concerning the research methodology are being further elaborated in the following sections.

3.2 Research approaches

Information systems research is at the crossroads of technical and human dimensions. Due to the latter, IS research is required to put emphasis on context. The organization is a prime *laboratory* for IS research, where the organization can take on different forms, such as commercial and non-commercial enterprises, or local communities (Braa and Vidgen, 1999).

IS is a highly applied field with strong vocational elements, for which a mix of practice and theory is needed to produce usable and relevant knowledge (Baskerville and Wood-Harper, 1996). IS researchers do not always remain passive observers, but may influence the local situation with the intention to learn from improvements to the problem situation.

Thus IS research has particular characteristics, like the challenge to understand the complex web of relationships in IS project contexts, and the possibly active involvement of IS researchers in the investigated projects. The wide range of phenomena in IS research can be studied from diverse perspectives. Accordingly, Nygaard (2002) suggested that for informatics research it is appropriate to add a fourth dimension, multiperspective reflection, to the usual set of three principal dimensions of sciences, observation, analysis and synthesis:

Observation: The empirical study of the phenomena: their identification, properties and behavior.

Analysis: Comprehension and explanation of phenomena in terms of an underlying theory.

Synthesis: Knowledge organized for the purpose of designing, generating or modifying phenomena.

Multiperspective reflection: The use of several perspectives in the consideration of phenomena, either from within the same science or drawn from more than one science; the study of how changes, introduced according to one viewpoint, affect properties of the phenomena when regarded from another viewpoint.

Depending on the aspects of interest, a single research perspective may be too restrictive (Orlikowski and Baroudi, 1991). In IS research, several major approaches, or

perspectives, have emerged (Chua, 1986; Gregg, Kulkarni, and Vinzé, 2001; Iivari, 2007; Orlikowski and Baroudi, 1991):

Behaviorist/positivist approach: Principal aim: Truth. It has its roots in natural science research methods and has been dominant in IS research, particularly in the English speaking world. It assumes that objective data can be collected and tested against prior hypotheses or theories. It attempts to generalize to larger populations with the use of statistical techniques.

Interpretive approach: Principal aim: Understanding. It is based on the belief that value-free data cannot be obtained, but researchers are inevitably influenced by their own preconceptions, and researchers furthermore interact with human subjects of inquiry, thereby influencing both researcher and researched subjects (Walsham, 1995a).

Critical approach: Principal aim: Emancipation. It questions the status quo and intends to lessen social inequality by analyzing barriers to development and by facilitating the improvement of social and economic conditions. This is done with the recognition of constraints in the given environment, including political, cultural and resource constraints.

Design science approach: Principal aim: utility. It is concerned with the process and product of socio-technical system design. Depending on the viewpoint, more or less emphasis is put on integrating social context into the design of artifacts (McKay and Marshall, 2005).

Positivist, interpretive and critical approaches have been identified as *paradigms* (Chua, 1986). Paradigms are fundamental sets of assumptions about knowledge (epistemology), how to acquire it (methodology), and about the physical and social world (ontology). Paradigmatic assumptions are shared by scientific and professional communities (Hirschheim and Klein, 1989). The assumptions are indeed so fundamental that a paradigm can be viewed as a set of basic beliefs, which must be accepted simply on faith. There is no way to establish their ultimate truthfulness. If there were, philosophical debates around paradigms "would have been resolved millennia ago" (Guba and Lincoln, 1994, p. 107).

Some propose that design science also be considered as a research paradigm, as an alternative to other paradigms such as the positivist or the interpretive paradigms (Gregg, Kulkarni, and Vinzé, 2001; Hevner et al., 2004; Kuechler and Vaishnavi, 2008b). In opposition to this view, McKay and Marshall (2007) argue that design science is more correctly seen as a body of knowledge, built through the application of a variety of research methods,

Table 3.1: Examples of research methods used in design science research, following positivist, interpretive and critical paradigms (McKay and Marshall, 2007)

Positivist	Interpretive and Critical
Lab experiment	Interview
Simulation	Participant and non-participant observation
Survey	Case study
Observation	Protocol analysis
	Participatory research
	Ethnography
	Action research

which may belong to any of the research paradigms. Thereby, a body of knowledge refers to the conflation of knowledge and method specific to IS design. In contrast, a paradigm does not include the knowledge that is generated by its application, but a paradigm consists of assumptions of ontology, epistemology and methodology, which guide the derivation of knowledge.

Without taking sides about the paradigmatic status of design science, it can be concluded that design science research methods extend to worldviews expressed by the three paradigms, positivist, interpretive and critical. Each of the three paradigms has particular strengths and weaknesses. Depending on the research objective, methods following solely the positivist, interpretive or critical paradigm on its own may not be able to fully cover the reality under investigation in IS research. Therefore, it has been suggested to combine methods of the different approaches. Design science is able to integrate methods of these paradigms. Table 3.1 lists examples of research methods following positivist, interpretive and critical paradigms that may be used in design science research. In addition to the methods in table 3.1, constructive methods for designing IT artifacts are essential for design science research (Iivari, 2007).

3.2.1 Positivism

Concerning epistemology, positivism clearly distinguishes facts and values. Scientific knowledge only consists of facts. The ontology of positivism assumes an objective reality that is independent of human interpretation (Walsham, 1995b). The methodology

for obtaining knowledge is context-independent and primarily quantitative (Gregg, Kulkarni, and Vinzé, 2001). Positivist studies aim to show fixed relationships between phenomena and to unveil truth or social laws. They are often used to test theory and to make predictive claims about phenomena (Orlikowski and Baroudi, 1991). The positivist paradigm still plays a strong role in IS research.

For example, DeLone and McLean (1992, 2003) have worked out causes and effects of success in a model for the measurement of IS success, which tries to impose some order on IS researchers' choices of success measures. To illustrate potential shortcomings of relationships based on positivist methods, the model does not recognize that different participants may have different conclusions about success of the same IS (Seddon et al., 1999). Positivist methods are thus not always the ideal choice for investigation into the complexity of the social world and some have voiced doubts concerning the usefulness of research results for practice (Frank, 2006), and some have cautioned against undue determinism based on causal associations:

> "While expected relationships may hold empirically for certain organizations in certain historical and socio-economic conditions, the ever-present ability of actors to alter the cycle of development, appropriation, institutionalization and reproduction of technology may undermine any causal relationships" (Orlikowski, 1992, pg. 34).

3.2.2 Interpretative approach

Interpretive studies consider reality as constructed by humans, either intersubjectively, like in the form of shared meanings, or with each person constructing his own reality (Walsham, 2006). The latter is an extreme standpoint, and as R. Weber (2004) points out, "surely some kind of reality exists beyond our perceptions of it!" (p. v). Independent of the chosen standpoint, the difference to the ontology of positivism is that in the interpretive approach multiple – socially constructed – realities exist (Gregg, Kulkarni, and Vinzé, 2001)

The interpretive approach considers facts and values as intertwined. In contrast to the epistemology of positivism, interpretive studies require the researcher therefore to make values explicit (Walsham, 2006).

The methodology for scientific knowledge construction in the interpretive approach is primarily qualitative (Gregg, Kulkarni, and Vinzé, 2001). Methodologies include hermeneutics and phenomenology (R. Weber, 2004).

Case studies following the interpretive approach hardly have enough cases to generalize statistically to greater populations. Nevertheless, generalizations are possible, even for single case studies (Yin, 2009), to the following types of generalization (Walsham, 1995b): (1) Development of concepts; (2) generation of theory; (3) drawing of specific implications; and (4) contribution of rich insight.

3.2.3 Critical research

Critical research is not solely associated with global inequalities, but it is a relevant element for many studies in the development country context. Since development informatics is related to globally disadvantaged communities, there is almost inevitably a critical element present in DI research. The following observations concerning critical research can be made regarding the research in this thesis:

1. All forms of research conducted in organizations carry with them an element of intervention in the local context. Therefore, a critical perspective always has certain relevance to in-context IS research. "The need for a critical perspective is an outcome of the belief that any intervention should have the intention of changing the problem situation for the better rather than for the worse" (Braa and Vidgen, 1999, p. 26).
2. The intention of the OPUS project is to build local capacity that can maintain and evolve the installed system after external consultants leave and external funding has dried up. If developing country based organizations indeed achieve the capability to design and evolve ICTs, then this represents a change of the status quo, undermining of the current state of affairs, in which technology producers are predominantly situated in the industrialized countries, whereas developing countries are in a receiver's position. Therefore, such a project has an implicit critical agenda.

However, despite the relevance of the critical viewpoint, it is not paramount in the sense that a particular group of dominant actors is targeted to give up a share of their power. Therefore it will not be used as the guiding paradigm in this study, since at the immediate level of intervention, i. e. the organizational level, the given power balance is not essentially being questioned. Hence, although every study has critical elements, and the OPUS project like many other development cooperation projects aims at empowerment of disadvantaged populations, such dynamics are not the primary focus of investigation within the research project. The main objective is to create a useful system, not to undermine

social structure.

3.3 Two goals: research and practical outcomes

IS research has been criticized for little practical relevance, and it has been suggested to apply research approaches that balance interests of researchers with those of practitioners. This has been suggested both for IS research in industrialized country contexts (Cole et al., 2005) as well as for development country contexts: "Without an appropriate understanding of both the theoretical and the practical dimensions of ICT4D we will not be able to help people implement changes that will be of practical benefit to them" (Unwin, 2009, p. 4).

Theory plays an important part in enabling fundamental understanding of reality. In this respect theory facilitates insight and description of reality. It is however not the objective of theory to obliterate understanding of reality, e. g. through the unnecessary usage of complicated vocabulary (Nuscheler, 2004, p. 20). The ultimate goal of theory building is not limited to a better understanding the world, but also to better inform practical interventions (Heeks, 2001). In this sense, theories can be helpful to accumulate and organize knowledge in a systematic manner, which facilitates improved professional practice (Gregor, 2006). Not least, the practical relevance of theory is also expressed in the popular quote by action research pioneer Kurt Lewin: "There is nothing more practical than a good theory" (Smith, 2001). Nevertheless, the practical relevance of much of IS research has been limited to date (Rosemann and Vessey, 2008).

In developing country oriented research, there is a danger of taking away data from its originating context in order to inform theory building by scientists mainly in industrialized countries, without giving back practical value to the organizations being researched. This contradicts the very idea of development informatics, which aims to contribute to local development. Not least, this represents a moral problem (Heeks, 2001). Therefore, the development country context especially calls for a balance of theoretical and practical outputs.

3.4 Contextual implications for the research approach

An IS is the application of technical artifacts in social contexts. This also applies to the OPUS project, which is being investigated in this study. Consequently, IS research is differentiated from e. g. computer science or software engineering by its focus on socio-technical artifacts. Thus, important elements to investigate are both the technical artifact and its utilization in particular contexts (McKay and Marshall, 2005, p. 5):

> "IS is fundamentally about human activity systems which are usually technologically enabled, implying that the context of design and use is critical, and that research paradigms, practices and activities must embrace such a worldview".

A useful way to integrate context into IS research is to view the organizational or social environment as a source of opportunities and constraints for technical innovation. Thereby, the inspiration taken from a particular context is used to improve development, implementation and exploitation of information systems.

But this remains a weak relationship between context and technological change, if the interaction being set in motion by technological change is not taken into account. It is also important to consider the organizational or social change that is unfolding when introducing technological innovation. Organizations change over time, and the introduction of information systems contributes to the change. This can be illustrated with the exercise of requirements engineering. Gathering organizational requirements at a particular time, for example at the beginning of an information system initiative, can not produce definitive requirements, because the requirements continue to constantly evolve (Avgerou, 2001).

Attempts towards a closer integration of technology and its social context have been made under the name of socio-technical systems (Bostrom and Heinen, 1977). Furthermore, theories have emerged that consider technological innovation in interaction with organizational or other kinds of social change. One such theory is the *duality of technology* (Orlikowski, 1992), which is related to structuration theory (Giddens, 1984) and puts a focus on change over time. Another example theory is *actor network theory*, which considers technology as one of the actors in networks (Latour, 1996).

Thus, a proper way to incorporate context into information system studies is to consider context and technology not as largely separate, but to take into account the change in the networks of people and institutions in which ICTs are being introduced. For analyzing processes of change during the application of ICTs, two dimensions can be

combined (Avgerou, 2001):

- A horizontal analysis of the sequence of events over time
- A vertical analysis of higher and lower levels of context, e. g. from the international level down to the group or even individual level

Although context is an important element, which is often emphasized in development informatics research, excessive attention to context can also have negative consequences. Individual solutions to each context practically prevents reuse, and hence, scalability; software systems are typically developed in order to be implemented not only in a single, but in a larger set of institutions. Therefore a trade-off is required between localization and standardization (Rolland and Monteiro, 2002).

3.5 Data analysis considerations

There exist two types of data analysis: Quantitative and qualitative data. The data used in a research project depends on the data available and generated during the research project and the researchers' objectives. For example, quantitative data is often available when the study is based on secondary data, such as statistics.

Both quantitative and qualitative data can be combined within a single research project. For both types of data, guidelines for rigorous analysis shall be followed. Whereas norms for appropriate quantitative data analysis are relatively well known and well documented, high quality analysis of qualitative data is less common (R. Weber, 2009).

3.6 Research methods in interpretive studies

Interpretative research is not synonymous to qualitative research. Qualitative research may or may not be interpretative. For example, qualitative case study research can be positivist (Yin, 2009) or interpretative (Klein and Myers, 1999). In the following, the focus is on research methods from an interpretive viewpoint.

According to an analysis by R. Weber (2009), researchers within the DI field primarily use five types of quantitative and qualitative research methods:

1. Acquisition of data from secondary sources
2. Surveys
3. Field studies (especially case studies)

4. Action research
5. Field experiments

Sometimes two or more of these methods are used together, e. g. field studies in combination with data from secondary sources.

In IS research, commonly used qualitative methods are ethnography, action research, grounded theory and case study. All of these qualitative methods examine phenomena in a natural setting and use similar data collection methods, but they differ in important ways (Dubé and Paré, 2001). Qualitative data sources include participant observation, interviews, documents and the researcher's impressions and reactions (Myers, 1997).

Ethnographic research comes from the discipline of social and cultural anthropology where an ethnographer is required to spend a significant amount of time in the field. Ethnographers immerse themselves in the lives of the people they study and seek to place the phenomena studied in their social and cultural context (Dubé and Paré, 2001).

Action Research (AR) is a combination of action and research, in other words a combination of practice and theory. AR is committed to the production of new knowledge by seeking solutions or improvements to *real-life* practical problem situations. One distinguishing feature of AR is, therefore, the active and deliberate self-involvement of the researcher in the context of the investigation, thereby influencing practical project outcomes (McKay and Marshall, 2001).

Grounded theory originated in the work of the sociologists Glaser and Strauss (1967). It is a research method that seeks to develop theory based on empirical data, which is systematically gathered and analyzed (Dubé and Paré, 2001). It facilitates the identification of patterns in data, which subsequently enables theory building (Fernández, 2005). Martin and Turner (1986) have characterized grounded theory as an inductive method.

A case study is an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident (Yin, 2009).

3.6.1 Action Research

AR aims to solve current practical problems while expanding scientific knowledge. In contrast to other research methods where organizational phenomena are studied with-

out changing them, the action researcher creates organizational change and simultaneously studies the process (Baskerville and Myers, 2004). Hence, action research is oriented towards collaboration between researchers and practitioners. AR is a cognitive process that depends on the interaction between the observers and those in their surroundings. There is a reactive process of stimulus-response at work; when an action is taken in a certain context, a response is recorded. This stimulus-response process provides a filter to identify relevant actions and build causal models. In the broadest sense action research resembles researchers conducting a field experiment on themselves together with others. Action research rests in an interpretive philosophical framework, as the changes triggered in a particular context are first and foremost valid in the specific context, and therefore need to be interpreted in the same context, before generalizations can be made (Baskerville and Wood-Harper, 1998). The following is a popular definition of AR (Rapoport, 1970, p. 499):

> "Action research aims to contribute both to the practical concerns of people in an immediate problematic situation and to the goals of social science by joint collaboration within a mutually acceptable ethical framework".

This definition not only points to the collaborative nature between researcher and client, but also to their different interests, and highlights potential ethical issues. This definition is however of general applicability, and not specific to information system research.

For IS, action research is believed to have potential value to increase the relevance of IS research. Action research is especially appropriate to address *how-to* research questions (den Hengst and de Vreede, 2004). According to Avison, Baskerville, and Myers (2001) this makes AR particularly well suited to study information systems development. The following is a list of four premises for action research specific to the field of information systems (Baskerville and Myers, 2004):

1. The purpose of any action needs to be planned beforehand. Also the theory must be explicit before the action is taken. Otherwise there is a risk that the action is purposeless.
2. There has to be practical action in the problem setting.
3. The practical action must inform theory; the theory needs to be adjusted according to the practical outcome of the action.
4. The reasoning and action must be socially situated. The action researcher needs to be a participant observer. There has to be a collaborative team involved in reasoning, action formulation and action taking.

The essence of action research is a two-stage process. First, the diagnostic stage consists of collaborative analysis of the situation by researchers and subjects. Second, the therapeutic stage involves collaborative change by introducing change and studying the effects (Baskerville and Myers, 2004).

Most forms of action research involve iteration at some level in the activities. Typical iteration patterns follow a variation of the basic two-stage process consisting of analysis and implementation. However, the degree of iteration varies between different forms. Some forms, such as *Canonical Action Research* (Davison, Martinsons, and Kock, 2004), use iteration as their primary organizing principle, and the set of research activities is repeated until the practical problem is solved. Canonical action research uses a cyclical process model consisting of five steps: Diagnosis, action planning, intervention (action taking), evaluation (assessment), and reflection (learning). On the other end of the spectrum there is the 'linear action research' form. In this form activities are not programmed to be repeated until a desired result is achieved. Specific steps do not necessarily have to be followed, only that the process progresses steadily from initiation to conclusion. An example of a linear action research form is 'process consultation', in which an outside consultant assists in organizational development. The consultant needs to transfer values and skills to the client in order to enable the client to improve its self-helping capabilities. Values include priority of long-term effectiveness over short-term output, and ongoing diagnosis over generalizations and principles. The process consultation includes logical areas – rather than a temporal sequence – for the consultant to work: (1) initial contact with the client organization, (2) defining the relationship, (3) selecting a method of work, (4) data gathering and diagnosis, (5) intervention, (6) reducing involvement, and (7) termination (Baskerville and Wood-Harper, 1998).

AR can be conceptualized as consisting of a research cycle and a problem-solving cycle, which inform each other to produce research outcomes as well as solutions to practical problems (McKay and Marshall, 2001). The research cycle provides input to the problem-solving cycle in the form of knowledge application. In turn, the problem-solving cycle feeds into the research cycle in the form of knowledge discovery (see figure 3.1). Although the two cycles are conceptually independent from each other, in practice problem-solving and research activities in these circles are often intrinsically related and difficult to distinguish. However, the analytic separation clarifies how knowledge is applied and discovered interactively between problem-solving and research activities (Chiasson, Germonprez, and

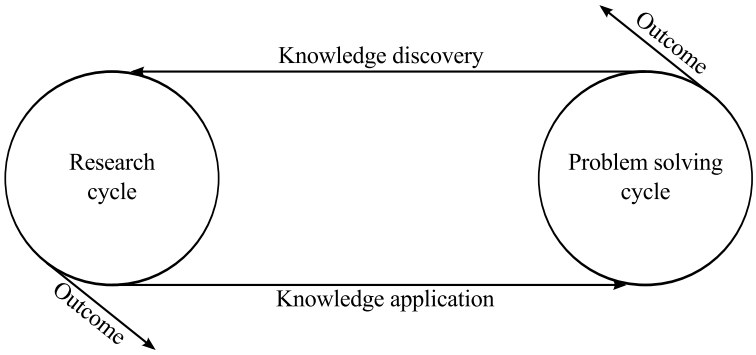

Figure 3.1: Problem solving and research cycles in action research (Chiasson, Germonprez, and Mathiassen, 2009)

Mathiassen, 2009).

Within the OPUS project, problem-solving was initiated first, and research activities were included slightly later, after the need had been recognized. Therefore, the knowledge-discovery aspect may be somewhat stronger than the knowledge application aspect. Chiasson, Germonprez, and Mathiassen (2009) refer to this constellation as the problem-solving dominant mixing approach to action research.

The overall applicability of AR for the Mozambican OPUS project can be characterized as follows:

- Action research can be considered useful for the Mozambican OPUS project because of the complexity of the context and the uncertainty of unfolding of events in the given context. Other research methods that use passive observation are less effective in filtering relevant actions, since they either require an *a priori* framework, such as a classification scheme, or an *a posteriori* framework such as grounded theory categorizations (Baskerville and Wood-Harper, 1998).
- The Mozambican OPUS project consists of a single high-level pass through the entire process of system production and introduction. Given the rather rigid time frame of development cooperation projects, the principal aim concerning practical project output is the development of a particular information system. Furthermore, an important goal in development cooperation projects is to nurture the local capacity and conditions towards self-help in the future. This represents an example of *linear action research*, similar to the *process consultation* model (Baskerville and Wood-Harper, 1998). Despite the linear nature, there is collaborative action between researcher and

other IS project participants, and research and practical outcomes are destined to influence each other. Ongoing participant observation can be expected to lead to an increasingly better understanding of the organizational problem and the possible solutions over time. Consequently, the research agenda is expected to evolve in response to this increased understanding.

- As an information system development project, different phases can be distinguished, such as definition, design, construction, installation and operation (Brandon, 2006). This is a way to identify smaller pieces within in the overall action research project.
- The project targets several universities in Mozambique. This already presents a certain plurality. In case the project has the chance of scaling up to other contexts, then further action research projects in different contexts may have the chance to strengthen insights gained during the Mozambican OPUS project (Braa, Monteiro, and Sahay, 2004).

In conclusion, the close observation and rapid intervention that is possible by AR make it a strong candidate to be integrated in the overall research approach.

McKay and Marshall (2001) make a distinction of the following elements in action research:

A: Problem situation of interest to the researcher.

P: Specific problem; a real-world instance of (A).

F: Theoretical framework.

MR: Research method, i. e. action research.

MPS: Problem solving method.

Prior to the intervention in A the research must declare a theoretical framework (F) and a method (M) which are used to guide the intervention and to make sense of the experience of the intervention. Subsequently, reflection takes place on these experiences, yielding findings about F, about M, about A and/or about the research themes. However, because of the dual cycle process there are two different M. One is the research method MR, which is Action Research. The other one is MPS, the problem solving method, which may or may not be an explicit method. Finally, P is a problem situation in the problem-solving cycle. P may be a specific, real-world example of any particular A, or it may be somewhat different, but allows the researcher to investigate A, so that there would be overlapping elements in P and A. The ownership of A rests with the researcher. By contrast, P remains in the ownership of participants or relevant stakeholders, and although the researcher is

Table 3.2: Action research variables in the OPUS project

F	Sustainability and how it relates to IS development, including processes of adaptation and standardization, and factors like utility, institutionalization and resources Knowledge as being the result of two-way communication and participation processes Iterative processes to increase the ties with the local context The potential of open source for learning and for higher education
M_R	Action research.
M_{PS}	A method for IS development following the system development life cycle concept.
A	Identify processes and factors to improve the success and sustainability of information system development projects with globally distributed project participants that possess widely varying degrees of skills, such as in the context of development cooperation projects.
P	Develop and implement the OPUS information system in the Mozambican higher education context in a sustainable way. Design a support structure. Put OPUS into open source tradition.

ethically bound to take an interest in P and to act to try to alleviate the problem, it is the participants who retain ownership throughout the research process. For the OPUS project the different variables are summarized in table 3.2.

3.6.2 Action research and sustainability

Sustainability is not only relevant to IS projects as such. It is also a topic in IS action research. Often, action research projects start locally. To be sustainable, efforts need to scale up from the pilot study in the initial context to other problem situations. In scaling action research efforts to other contexts, the influence by the researchers typically decreases (Braa, Monteiro, and Sahay, 2004).

As an outcome of a large-scale AR project, Braa, Monteiro, and Sahay (2004)

emphasize the importance of networks for action research projects and suggest an approach called *networks of action*, which intends to improve sustainability and is characterized by:

1. Abandoning single site action research projects in favor of a network of sites;
2. Generating local, self-sufficient learning processes with exchange of experiences between sites;
3. Nurturing a robust, heterogeneous collection of actors with sufficiently similar agendas;
4. Aligning interventions with surrounding configurations of existing institutions, competing projects and everyday practices.

3.6.3 Case study

A case study is characterized by Yin (2009) as an empirical enquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident. In contrast to action research, it does not attempt to create change during the investigation. The case study approach provides, at a first level, a descriptive model. More specifically, for information system projects it can be used to identify the key factors of IS projects beyond the IT system boundaries, i. e., the antecedent conditions, forces of change, forces of alignment, sequence of events and decisions, and outcomes over the course of the implementation (Montealegre, 1999).

In the case study method, comparisons can be made by combining literature with own findings. Generalizations can be drawn from empirical data towards theoretical propositions. This is also true for a single-case study. However, another type of generalization, statistical generalization towards greater populations, is not feasible based on a single or a few cases only (Yin, 2009).

Case studies can be used to achieve various aims: to provide description, to test theory or to generate theory. Theory-building from case study research is particularly appropriate when there is still little known about a phenomenon or there is not enough literature to engage in incremental theory building (Eisenhardt, 1989).

Case studies may be integrated with action research studies, either in a sequential manner, or with action research being the dominant method, whereby case studies are being used to clarify certain research questions along the way (Chiasson, Germonprez, and

Mathiassen, 2009). For example, a case study may be executed at the step of evaluation in action research dominant studies.

Another form of combination has been suggested by Braa and Vidgen (1999) who use the term *action case* to refer to studies that are neither pure case studies nor full blown action research studies. Rather than pure interpretations by a detached, outside observer, such studies may involve a certain element of change, for example by bringing together people from different parts of an organization, particularly where no formal communication channels exist currently, or by distributing research reports to management staff.

Concerning the research within the OPUS project, the case study method can be a useful addition to an action research dominant mode of investigation. Due to the ongoing element of change created by the researcher, the case study method on its own is not ideal to provide a realistic approach to the entire research project. Action research is a more realistic model, which is also more likely to positively influence the practical project outcome.

3.7 Design science and design research

Design science has its roots in engineering and in the sciences of the artificial (Hevner et al., 2004; Simon, 1996). It has a long tradition in Europe, being particularly dominant in the German speaking countries (Lange, 2006; Wilde and Hess, 2007), albeit often in an implicit manner, i. e. without using the term *design science research* for the applied research methodology (Niehaves, 2007). Examples for design science research include the activities that many computer scientists have been doing right from the beginning: developing computer architectures, programming languages, algorithms, database management systems and many other innovations. However, IS research has somewhat lost sight of its design science origin (Iivari, 2007).

Design science is relevant for IS research because the field should not only try to understand the world, but should also change it (Carlsson et al., 2011). Understanding, which is facilitated by research approaches other than design science, is only halfway to solving problems in IS. Understanding identifies yet unresolved problems. The next step is to develop and test alternative solutions. Therefore, in addition to description-driven research, there is a need for prescription-driven research to guide managers in designing solutions (van Aken, 2004). Such prescription-driven research shall, however, not be confused with

consulting; not the application of scientific knowledge to solve a particular practical problem is the focus of prescription-driven research, but the development of scientific knowledge to solve a *class of problems*, in other words, the development of abstract knowledge (McKay and Marshall, 2005; van Aken, 2004). This shall be done not by presenting recipes, but through the development of "field-tested and grounded technological rules to be used as design exemplars of managerial problem solving" (van Aken, 2004, p. 221). Design knowledge occupies the middle ground between descriptive theory and actual application: "A design-science is not concerned with action itself, but with knowledge to be used in designing solutions, to be followed by design-based action" (van Aken, 2004, p. 226).

The relevance of design science is increasingly being recognized, also within the United States based research community: "[O]ur focus should be on how to best design IT artifacts and IS systems to increase their compatibility, usefulness, and ease of use or on how to best manage and support IT or IT-enabled business initiatives" (Benbasat and Zmud, 2003, pp. 191).

There is a further aspect about the design perspective: On the one hand, critique is frequently expressed about overemphasizing technological solutions, for example, but not exclusively, in DI initiatives (Boyle, 2002). This indicates a tendency by practitioners to focus on technology rather than context. But on the other hand there is little IS research that avoids black-boxing and looks in detail at design aspects of the artifacts (Orlikowski and Iacono, 2001). By theorizing about design process *and* product (Gregor and Jones, 2007), the design science research approach has the potential to overcome this situation and produce rigorous and relevant results (Niehaves, 2007) and to avoid black-boxing either context or technology.

3.7.1 Design research in relation to other IS research approaches

Design research does not contradict with other research approaches. It is compatible with positivist, interpretive or critical epistemologies. The chosen epistemology has strong impact on the evaluation of design research results (Niehaves, 2007). For example, March and Smith (1995) and Hevner and March (2003) suggest a twofold IS research cycle. One part of the research cycle consists of the design of artifacts based on existing theories. Complementary, the second part uses behaviorist science methods for testing the artifacts and the accompanying new or improved theories.

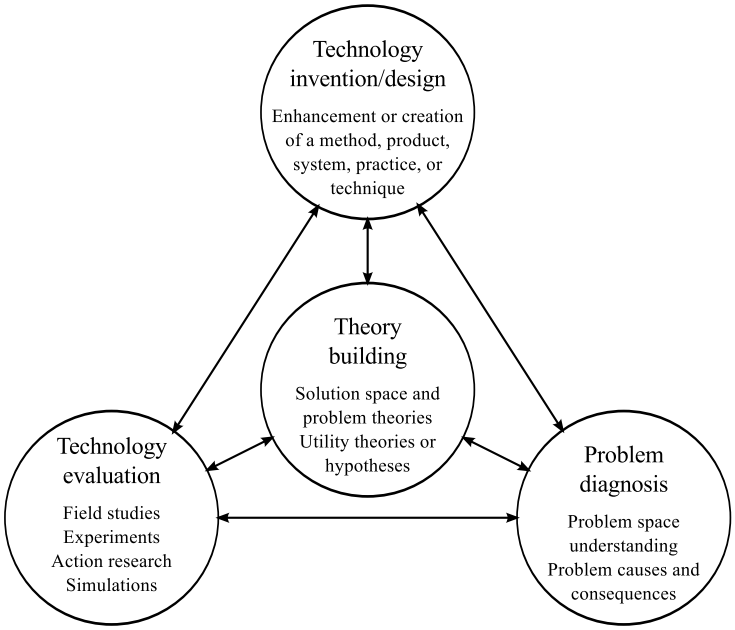

Figure 3.2: Activity framework for design science research (Venable, 2006)

Venable (2006) goes further and puts theory building based on design research in a broader context. Theory building is a design science activity that doesn't work in isolation, but is interlinked with (a) problem diagnosis, (b) technology invention or design, and (c) technology evaluation (see figure 3.2). Thereby, "problem diagnosis and technology evaluation may be undertaken in the empirical domains of natural and particularly social/behavioural sciences" (p. 16). Theory building is the link between all the other activities.

An essential method in design science research is the construction of IT artifacts. This is however an activity, which is not in the exclusive domain of researchers, but is also carried out by practitioners. Two options exist to distinguish design activities of practitioners and researchers. One is the scientific evaluation of IT artifacts. A second option is to specify a rigorous research method for building IT artifacts (Iivari, 2007).

3.7.2 Relationship between design science and action research

Both action research and design science are proactive and do intervene in the real world. They do not limit themselves to studying phenomena as distant observers. Therefore, certain overlaps can be identified between the two approaches. Both have research cycles

Table 3.3: Action research (Susman and Evered, 1978) and design science activities (Kuechler and Vaishnavi, 2008a)

	Action research steps	*Design science steps*
1	Diagnosing	Awareness of problem
2	Action planning	Suggestion
3	Action taking	Development
4	Evaluating	Evaluation
5	Diagnosing	Conclusion

that consists a similar set of main activities (Iivari and Venable, 2009; Järvinen, 2007), as illustrated in table 3.3.

Due to the observed overlaps, Järvinen (2007) goes as far as to suggest action research and design science be considered as essentially similar research approaches as far as IS research is concerned. He argues that action researchers, just as design science researchers, have an interest in the utility of the system, and in cooperation with the practitioners they plan the desired future system. In contrast, other methods, such as interpretative and positivist methods are not concerned with the goal function, i. e. the utility of the system. Rather, these methods can be used to describe and analyze existing systems, both before and after changing the system. This argument leads to the proposition that action research and design science differ from natural and social sciences, and that action research shares more similarities with design science than with other positivist or interpretative methods. Hence, Järvinen (2007) claims that action research and design science should be considered as similar research approaches.

However, others are more cautious about the level of similarity between action research and design science. For example, Cole et al. (2005) recognize that design research and action research make different kinds of theoretical contributions. Design research makes theoretical contributions more in the form of the construction of artifacts, whereas action research focuses on adopting technology rather on than building it. Furthermore, in many cases, design science research performs construction and initial evaluation (testing) of artifacts in a laboratory environment, not directly at client sites (Iivari and Venable, 2009).

Iivari and Venable (2009) call the conclusions by Järvinen (2007) as “overly hasty”.

Table 3.4: Overlaps in activities between action research and design science research (Iivari and Venable, 2009)

Case	AR interest	DSR interest	DSR activities
1a. No overlap	Understanding reality in an organizational context	None	None
1b. No overlap	None	Solving a purely technical problem by developing and evaluating a new solution technology	Theory building, solution technology invention, and artificial evaluation
1c. No overlap	None	Solving a socio-technical problem in a non-AR context by developing a new solution technology, but evaluating it by means other than AR	Theory building, solution technology invention, and artificial and/or naturalistic evaluation
2. Slight overlap	Evaluating an existing solution technology in an organizational context	Evaluation of a solution technology developed separately	Naturalistic evaluation only
3. Significant overlap	Solving a socio-technical problem by developing a new solution technology and evaluating it in an organizational context	Solving a socio-technical problem by developing a new solution technology and evaluating it in an organizational context	Theory building, solution technology invention, and naturalistic evaluation

According to their critique, the similarities only apply to certain types of action research and design science, but it would be simplistic to interpret all design science research as action research. Based on an analysis including paradigmatic assumptions, Iivari and Venable (2009) conclude that action research and design science research sometimes, but not always, share the same paradigmatic background, and that there may be no, little, or significant (but not total) overlap between action research and design science research (see table 3.4. On the other hand, the potential overlap means that design science and action research are not mutually exclusive. In certain cases, design science research and action research can indeed be similar.

The case of the OPUS research project represents a significant overlap of AR and Design Science Research (DSR) interests. There is a socio-technical system to be developed in the context of several universities, and the IS development process is to be evaluated. Participant observation is an essential part of the investigation. The goal is to identify process elements and models such as a feasible support structure. Continuous evaluation and feedback is vital to keep focused. Both AR and DSR have particular strengths to

contribute to this task.

But a few more remarks are appropriate concerning the synergistic use of AR and DSR in the case of the OPUS project. Even in the case of significant overlap between AR and DSR interests, Iivari and Venable (2009) are reluctant to jointly apply AR and DSR approaches, because in their argument artifacts that are designed in DSR projects shall first be tested in laboratory conditions, before being evaluated at clients. Otherwise clients may be misused as *guinea pigs*. Therefore, testing technology 'on the fly' through AR activities requires special care and shall only be done in special circumstances. The OPUS project may indeed represent such special circumstances, in which new technology is introduced in a very different context than what the majority of developers are used to. Furthermore, according to Iivari and Venable (2009), AR activities are particularly useful in improving technologies. Therefore, the AR outcome may well serve as input to design science, to illuminate the design for the class of problems that consists of developing IS with participants of widely varying backgrounds. The following section on design theory provides a possible foundation for the formulation of design knowledge that can draw on insights gained in the course of an action research project.

3.7.3 IS design theory

Theories for design and action are being highly influential in IS, even though they are not always recognized as theories. Examples for design theories include the relational database model (Codd, 1970, 1982) and the Systems Development Life Cycle (SDLC). Gregor and Jones (2007, p. 313) bluntly state that "it is difficult to over-emphasize the significance of design work and design knowledge in IS for both research and practice".

A design theory, as it is used here, shows the principles underlying the design of a certain IS artifact, which are based on knowledge of both IT and human behavior. The word design is both a verb and a noun, and accordingly, a design theory can have either a method (e. g. the SDLC) or a product (e. g. a database or a decision support system) as its primary design goal. The word artifact designates something that is artificial, or constructed by humans, as opposed to something that occurs naturally (Simon, 1996). Both design methods and products are called artifacts.

Design theories are abstract, and include other abstract ideas such as models and algorithms. An instantiated design theory has, however, a physical existence in the real

Table 3.5: Five types of theory (Gregor, 2006)

Type	Name	Characteristics
I	Theory for analysing	Analytic theory to describe *what is*, without the power to explain causal relationships or make predictions; the most basic type of theory. Needed when nothing or very little is known about the phenomenon in question. Taxonomies are a variation of this theory type.
II	Theory for explaining	To explain how and why certain phenomena occur. The primary goal is not to predict the future, sometimes not even to generalize, but to enlighten and to show a way to view the world. Focus on understanding. One example is structuration theory. No testable propositions.
III	Theory for predicting	Says what will be, but now why, by showing relationships between independent and dependent variables. Parts of the system are a *black box*. Statistical techniques are often used. Not very common in IS research. Testable propositions exist.
IV	Theory for explaining and predicting	Says what is, how, why, when and what will be. Provides predictions, testable propositions, and explanations. Corresponds to commonly held views about theory in natural and social sciences. One example is the DeLone and McLean model of IS success.
V	Theory for design and action	How to do something. Appears under different labels, for example constructive research, software engineering research, prototyping, and design science. Important theory type in IS. Action research is seen as particularly appropriate approach for this theory type.

world (see figure 3.3). In opposition to Hevner et al. (2004), Gregor and Jones (2007) suggests that constructs, models and methods are considered theory or components of theory. Theory is explicitly not seen as something that would exclusively belong to natural or social science. Gregor (2006) has distinguished five types of theory relevant to IS (see table 3.5). One of them, type V, *theory for design and action*, corresponds to abstract artifacts as displayed in figure 3.3. For this type of theory, action research is considered as a particularly relevant research approach.

As shown in figure 3.3, instantiations have a physical existence in the real world in the form of a piece of hardware, software or physical actions. Theories themselves do not have a physical existence. Theories include constructs, methods and models. Human understanding is the basis for the observation of instantiated artifacts and subsequent conceptualization in abstract, general terms. In turn, theories are used to guide the building of instantiations in the real world.

Walls, Widmeyer, and Sawy (1992) made an early attempt to guide the formulation

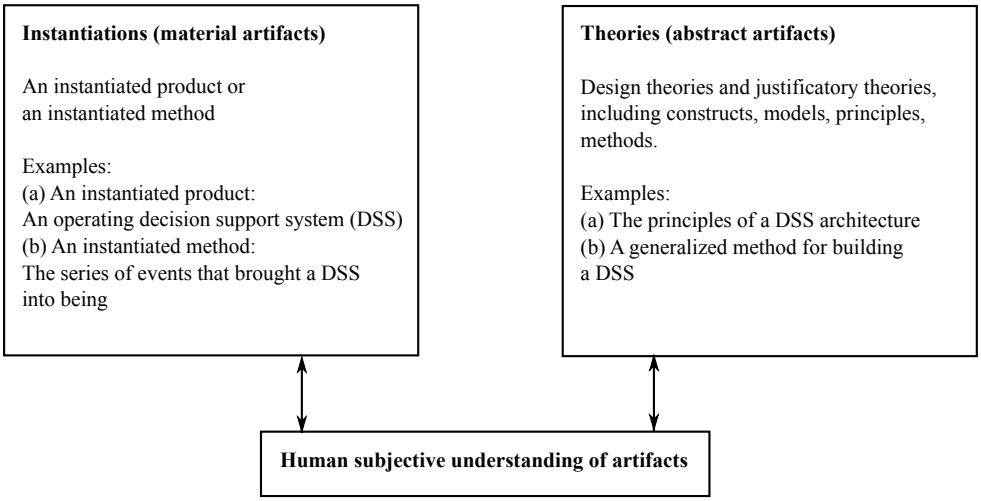

Figure 3.3: Relationships among IS/IT artifacts (Gregor and Jones, 2007)

of an IS design theory, adapting the ideas from Simon (1996) about artificial sciences to IS. They make a distinction between product and process and propose the specification of meta-requirements that represent a class of systems. The meta-requirements are to be met by a meta-design, which consists of a set of artifacts. The proposal also includes a design method that describes how to construct the artifact; the design theory is to be based on kernel theories, with separate sets of kernel theories for product and process, respectively. The authors argue for testable hypothesis for product and process.

Gregor and Jones (2007) extend the specification of Walls, Widmeyer, and Sawy (1992) in several ways. First, they add *constructs*, *artifact mutability*, and an *expository instantiation* as possible components of a full ISDT specification. Furthermore, they simplify justificatory knowledge into a single component, instead of the two distinct sets of kernel theories for product and process. The resulting set of eight components has six core components that any Information System Design Theory (ISDT) specification should provide, and two further optional components (see table 3.6). The six core components are sufficient for a thorough description of the artifact. In addition, the expository instantiation may provide a proof of concept and can be added later. However, if the design theory is developed within an action research project, the real world implementation is an implicit part of the theory development process.

Table 3.6: Eight components of an Information System Design Theory (ISDT) (Gregor and Jones, 2007)

	Component	*Description*
	Core components	
1	Purpose and scope	What the theory is for. The set of meta-requirements or goals that specify the type of artifact that the theory applies to.
2	Constructs	The entities of interest in the theory
3	Principle of form and function	The abstract blueprint or architecture that describes an artifact, i. e. either a method or a product
4	Artifact mutability	The changes in state that are anticipated in the theory, that is, what degree of artifact change is encompassed by the theory
5	Testable propositions	Truth statements about the design theory
6	Justificatory knowledge	The underlying knowledge or theory from the natural, social or design sciences that serves as a basis and explanation for the design (kernel theories)
	Additional components	
7	Principles of implementation	A description of processes for implementing the theory (for either a product or a method) in specific contexts
8	Expository instantiation	A physical implementation of the artifact that can assist in representing the theory both as an expository device and for purposes of testing

3.8 Summing up the research methodology

The characteristics formulated at the beginning of this chapter (see section 3.1) translate into the following considerations regarding research methodologies:

Action research is the most convincing candidate as the dominant research method. The complexity of the context calls for participative observation and actively creating change in order to improve the practical project outcome. Action research also helps to conceptually distinguish problem-solving and research interests.

Case study: The context of IT implementation in a developing country and the longitudinal type of investigation suggest case study research as a possible option. However, action research as the principal research method better reflects the reality. But a case study may be incorporated into or follow the action research project, e. g. to test theory generated by action research.

IS design theory: The insights gained during the IS innovation effort can be usefully summarized with an IS design theory. It incorporates knowledge about artifacts as well as justificatory knowledge and insights gained through methods such as action research and case studies. It is a way to integrate knowledge about both context and technical details, thereby avoiding to black-box either of them.

Table 3.7 outlines the research in this thesis as presented in the subsequent chapters. Chapter 4 on appropriate technology establishes guiding principles for further investigations. This is done by doing a case study that analyzes how appropriate technology principles fit with information system development, based on experiences in the beginning of the OPUS project. It is concerned with providing context-sensitive solutions and is of primary importance for early phases like the problem definition. Subsequent chapters are based on action research, that is, by trying to solve practical problems at the same time as gaining theoretical insights. Open source plays a major role in the construction of technical artifacts. It facilitates the collaboration between globally distributed actors (chapter 5). Organizational implementation of information systems is related to managing organizational change (chapter 6). For the further operation and maintenance of information systems, support is a prerequisite (chapter 7). The structurational analysis again uses a case study method to gain an overall understanding of the entire project life cycle (chapter 8). Finally, design science is used to synthesize the insights are into an IS design theory (chapter 9).

The different parts of the research project can be put into perspective of the four dimensions put forth by Nygaard (2002): observation, analysis, synthesis, and multiperspective reflection. The chapter on appropriate technology lays some groundwork by making observations of the events in the OPUS project and the practice of software development in relation to appropriate technology principles. These observations are then analyzed to make conclusions concerning cross-cultural information system development. The chapters on appropriate technology, open source, change management and support have a practical orientation to solve real world problems encountered in the OPUS project. Therefore, in addition to observation and analysis, there is a focus on synthesis to formulate building blocks for information system projects. The structurational analysis pays tribute to multiperspective reflection. The insights from previous chapters, such as the derived building blocks, are considered from a different angle; using the sociological theory of structuration, this chapter analyzes how events were perceived by the different participants and which potential and actual conflict emerged. All of the insights are finally synthesized into the *Appropriate Information System Development (AISD)* methodology, which is the primary outcome of the research in this thesis. The AISD methodology is an example of an IS design theory.

Table 3.7: Research outline

	Chapter	*Focus*	*SDLC phases*	*Research method*
4	*Appropriate Technology*	Local context and relevance	Definition, design	Case study
5	*Open source*	Collaborative development	Construction	Action research
6	*Managing change*	Organizational implementation	Implementation	Action research
7	*Support / empowerment*	Community based support	Operation and maintenance	Action research
8	*Structurational analysis*	Understanding project success	All phases	Case study
9	*AISD methodology*	Integrate insights into a methodology	All phases	Design science (IS design theory)

Part II

Investigation of the research questions

Chapter 4

Appropriate technology: How can ISD deliver meaningful solutions to local problems?

This chapter has two major goals. First, the concept of *Appropriate Technology (AT)* is reviewed in order to investigate what can be learned for information system development, to note particular challenges of information system projects in relation to appropriate technology principles. This analysis is the basis for further theoretical model developments throughout the thesis.

A second goal in this chapter is to critically evaluate the so-called *Appropriate ICT (AICT)* framework with the Mozambican OPUS project. The insights gained from this evaluation are a concrete starting point for the development of a methodology for collaborative inter-organizational information system development, which is further elaborated in the subsequent chapters.

The first section outlines the concept of appropriate technology. Next follows an analysis of the compatibility of software development with the appropriate technology philosophy. This is done by looking at the characteristics of software and open source in relationship to AT principles, which results in potential opportunities to bridge the digital divide. This is followed by an introduction to the Appropriate ICT framework, which is derived from appropriate technology guidelines. Subsequently, the AICT framework is evaluated and possible extensions are suggested. This is done by comparing actual events

of the OPUS project with issues emphasized by the AICT framework, more specifically tool usage and guiding questions.

4.1 Appropriate Technology

Appropriate technology is an approach to technology construction and use that aims to be suitable for the social and economic conditions of the local community into which the technology is being introduced. It can be broadly conceptualized as arising out of the necessity to make use of the available means to satisfy the elementary needs of the community (Tharakan, 2006). An overall objective of appropriate technological choice would be the achievement of greater technological self-reliance and increased domestic technological capability, together with fulfillment of other developmental goals (Kahen, 1995).

Appropriate technology is a way of thinking about technological change and has its roots in *intermediate technology*, which was envisioned by Schumacher (1973) as a technology for developing countries that is immensely more productive than indigenous technology and at the same time immensely cheaper than the sophisticated, highly capital-intensive, technology of modern society. He reasoned that the gap between traditional and modern technology is too big for a smooth transition from one to the other, and the infiltration of modern technology would kill traditional technologies and workplaces faster than modern workplaces could be created.

According to McRobie (1979) the essence of Schumacher's ideas on intermediate technology is that the increasingly complex and costly technologies of rich countries are generally inappropriate for the poor, and that the choice of technologies is the most critical choice confronting any developing country. By distinguishing science and technology, McRobie outlines the many possible faces of technology, depending on the particular circumstances:

> The knowledge of scientific laws . . . is, in a sense, absolute, and one could hardly talk of intermediate knowledge or intermediate science. But the application of the best knowledge can take many forms and lead to many types of technology and modes of operation. It is here that the need for, and the possibility of, intelligent choice enters. (McRobie, 1979, p. 72)

The pursuit of appropriate technology does not necessarily mean the employment and adoption of low level technology (Kahen, 1995). However, the historic concept of intermediate technology as being located between the traditional technologies of underdeveloped

and the high tech of developed economies, has led to the occasional misconception of intermediate technology as necessarily being low-tech (Tharakan, 2006). Hence, the term *intermediate technology* was criticized for being suggestive of inferior technology, and "for implying a technological fix for development problems, separate from the social and political factors involved" (Hollick, 1982, p. 214). The name *appropriate technology* was suggested as a substitute, in part to better account for the social and cultural aspects of innovation. It has evolved into a development approach that is aimed at tackling local community development problems, and is based on the belief that communities shall be involved in deciding how their future will look like and which tools and techniques to use to reach their goals. It emphasizes local needs and values. It tries not only to make optimum use of existing skills and resources, but also to build skills and resources to raise the productive capacity of a community (Akubue, 2000).

The appropriateness of technology can be determined through its acceptability, adoption and institutionalization in a new organizational setting (Kahen, 1995). However, there is no definite, commonly agreed set of criteria that appropriate technology projects are supposed to follow. As an illustration of the practice of appropriate technology projects, the following is a list of guidelines for a wide variety of appropriate technology oriented innovation (Darrow and Saxenian, n.d. van Reijswoud, 2009):

1. It should be possible to implement/realize technological solutions with limited financial resources.
2. The use of available resources must be emphasized in order to reduce costs and to guarantee the supply of resources, e. g. for maintenance.
3. Technologies may be relatively labor-intensive, but must have a higher output than the traditional technologies.
4. The technology must be understandable for people without specific or academic training.
5. Small rural communities should be able to produce and maintain the technology.
6. The technology must result in economic and/or social progress.
7. The technology must be fully understandable for the local population, the end-users, resulting in possibilities for them to become involved in the possible innovation and extension of the use of the technology.
8. The technological solutions must be flexible and easily adaptable to changing circumstances.

9. The technology must contribute to an increase in productivity.
10. The technology should not have a negative impact on the environment.

Despite the decline of the appropriate technology movement in recent years (Polak, 2010), valuable lessons have been learned concerning technology in developing countries, which have relevance for the application of ICTs in this context, particularly the integration of the local community into the technology innovation process. These lessons on the part of appropriate technology in developing countries have not yet been fully integrated into research and practice of ICT projects (Heeks, 2008).

Information technology has been identified as an appropriate technology. Computers are relatively inexpensive and they are user-friendly with a low level of skills requirements. This is true despite the fact that computers are not simple to build, to program and to maintain. They promote decentralization and do not need the sophisticated infrastructure of modern industrialized countries in order to function. A particular challenge, however, remains with the adaptation of IT based systems to end users in developing countries, because IT systems that are imported from industrialized countries have been designed to solve problems in the context of origin (Kahen, 1995). For example, computers are not built for high temperature and humid environments, which are common in many developing countries.

4.2 Information systems and appropriate technology

Often, the production and use of information systems depends to a large extent on its software component. Therefore it is pertinent to make some observations about software in relation to appropriate technology principles:

- IS development projects are inherently complex, because they deal not only with technological, but also with organizational factors, often beyond the project team's control (Xia and Lee, 2004). In developing country contexts, this becomes particularly apparent. Nauman, Aziz, and Ishaq (2005) provide a case study of the complexity of a software development project in a developing country in which they demonstrate considerable complexity in a seemingly simple information system project.

  In order to attend to local circumstances, AT artifacts need to be simple to use and simple in routine maintenance, but do not necessarily have to be simple to design. For software artifacts this translates to a need to emphasize usability and to hide

complexity from users and technical maintainers in their routine work.

- Because they are complex and need a lot of resources to develop, software systems should not be reinvented by each organization that is in need of similar systems. In contrast to the development of hardware artifacts, it is relatively easy to collaborate in software development projects via the Internet. Uwadia et al. (2006) describe a case study of the collaborative development of information systems in the university environment of Nigeria. The collaborative approach was viewed appropriate for Nigeria because of a high degree of similarity in the core functions and activities needed to support the administration and management of the universities.

  However, collaborative work without face-to-face contact is rather uncommon in the developing country context and presents an additional challenge (Favela and Peña-Mora, 2001). Another issue is the fact that benefits for the community involved may not be visible until late in IS development projects, due to the long process of system development. Therefore, proper ways of organizing collaborative software development need to be investigated and developed.
- For any technology there is a design challenge to adapt to specific local needs concerning the technology use. Software systems can be more flexible than hardware artifacts regarding local customization. To take advantage of this possibility, the involvement of the local community in the collaboration is important. The participation of local users and developers throughout the system development life cycle can help in designing a system with beneficial customization options.
- Software needs constant, ongoing development to respond to changing organizational needs (Lehman and Ramil, 2001). Therefore, local capacity has to be created to enable continuous development, e. g. by finding ways to involve local participants such as software developers early in the SDLC stages.

These observations show the importance of community networking and capacity building, and it follows that further tools and methods should be investigated, for example to effectively coordinate inter-organizational collaborations for the development and maintenance of ICT systems. Therefore in the following some conclusions will be drawn about how distributed software development, particularly from the point of view of open source, fits with developing appropriate ICT.

Software has several characteristics that fit in well with the AT concept. Software production needs small amounts of initial capital, since computers and Internet connections

are the only investment. It is labor-intensive and has the potential to offer more productive solutions than traditional information management technologies such as pen and paper and plain office applications like word processors and spreadsheets. Moreover, collaborative software development offers possibilities for local experts and entrepreneurs to get involved if enabling conditions are in place; open source software implicitly assumes that people work together to achieve overlapping goals.

One of the conditions for success for business critical information systems is that the system is considered sufficiently reliable, not error-prone. This is relevant for the user acceptance of the system. To maintain the code base constantly in good shape requires experienced software developers. This is a point of concern, given the AT principle that the technology should be maintained by local stakeholders without the need for external expertise.

The high level of skills required to develop and maintain the code base of a software project brings with it some serious challenges for sustainability and support. In the OPUS case, no single university is likely to have both the capacity and the acceptance of the other universities to maintain such a software system on its own. This makes the question of appropriate support and sustainability a crucial one. To build support capacity, participation of local participants in open source projects can be one of a number of skills development pathways.

To summarize, AT and open source software development fit well in many respects, but an enabling environment is necessary to create a stable project situation and overcome potential sustainability issues. Local competencies need to be built up carefully over an extended period of time.

4.3 Appropriate ICT

Appropriate technology has been applied in many domains, such as architecture, building, energy and water supply (Darrow and Saxenian, n.d.). There have only been few appropriate technology initiatives in the domain of ICT. The following lists some examples of projects with varying degree of success to develop appropriate hardware or software solutions for the developing country context:

One Laptop per Child (OLPC) (`http://laptop.org`) is a project, backed by UNDP, to provide technology such as laptops to schools in least developed countries. The

laptops are sold to governments, which then distribute *one laptop per child*. The operating system and software is localized to the respective countries. The goal was to provide laptops for the price of $100, but the price remained well above this goal. However, several million laptops have been shipped to developing countries.

Simputer (`http://www.simputer.org`) was a Linux based handheld computer developed by Indian non-profit organization consisting of scientists and engineers. It was primarily distributed in India. It was envisioned as a low-cost alternative to personal computers. There were, however, not as many Simputers sold as intended.

Damn Small Linux (`http://www.damnsmalllinux.org`) is a Linux distribution that makes minimum requirements on hardware. It is small enough to run on old 486 processors. Moreover, it can run completely inside a computer's memory. It may boot from CD, USB or even from within another host operating system such as Windows.

The aspect of organizational change has not received much attention, neither in appropriate technology in traditional domains nor in the ICT hardware and software projects listed above. For example, the criteria by Darrow and Saxenian (p. 113) highlight the product design, but are less concerned with the process of introducing the product into a community, i. e. the organizational implementation process. As shown in the following, *Appropriate ICT* takes a closer look at the complexities of ICT projects and integrates organizational change.

Appropriate ICT (AICT) applies the concept of appropriate technology to ICT projects. In accordance with AT, it emphasizes local communities and technological change management. AICT refers to practices in the field of community informatics to address the process of technological change in developing ICT systems. Appropriate ICT is described as "the integrated and participatory approach that results in tools and processes for establishing Information and Communication Technology (ICT) that is suitable for the cultural, environmental, organizational, economic and political conditions in which it is intended to be used" (van Reijswoud, 2009, p. 6).

An immediate consequence of this definition is that this approach (1) combines stakeholders from a broad perspective and (2) is organized around the community in which the software product is to be used. Therefore this approach is promising when it comes to bridging a digital divide, particularly when there are international stakeholders. In this case the approach may lead to balancing differences. It is even more effective when the application has a broader focus, and thus involves larger groups of people.

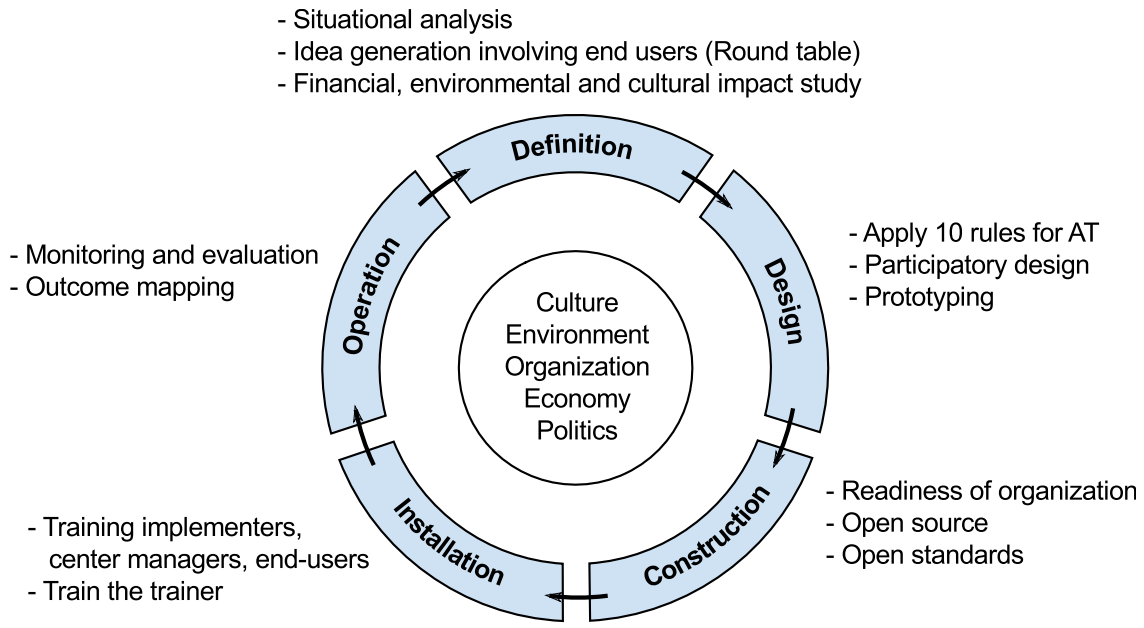

Figure 4.1: Foundation of the Appropriate ICT framework and supporting tools and methods for ICT production and use (van Reijswoud, 2009).

The theoretical basis of the Appropriate ICT framework consists of the distinction between the product, i. e. the technical artifact itself, and the process of introducing this artifact into the target organization or the respective environment. The process perspective is vital during the installation phase and is guided by community informatics practices, involving the community itself in the adaptation of the ICT artifact.

The Appropriate ICT framework is based on a flavor of the traditional Systems Development Life Cycle (SDLC) and extends it with tools to guide the ICT solution to greater appropriateness and thus effectiveness in implementation (see figure 4.1). The AICT framework proposes a possible set of tools, but also encourages the use of further tools that address cultural, environmental, organizational, economical and political aspects of ICT projects. Additionally, a set of key guiding questions, which are structured along the phases of the SDLC, integrate AT principles and serve to reflect on the three aspects of hardware, software and organizational change (see table 4.1).

Table 4.1: Key guiding questions for Appropriate ICT development (van Reijswoud, 2009)

Hardware	Software	Change management
	Definition phase	
Specific requirements to hardware in terms of climatological and environmental conditions? What are the possibilities in terms of enabling factors (Internet connectivity, electricity)?	What are the needs? What are the expectations?	What ICT knowledge levels? What the financial constraints? What is the cultural context? What added value is created? How is the economic equilibrium affected? What new ways of working are introduced? What will the impact be of the system in terms of organizational change? What is the involvement in the idea generation of key decision makers (political leaders, religious leaders)?
	Design phase	
What is offered on the local market? What are physical constraints? What the financial constraints?	What interoperability needs? What localization is needed? What flexibility is expected?	What are the information needs of the various target groups? How will these needs evolve? How do the expectations change?
	Construction phase	
What local skills are available? Is the equipment protected against physical conditions?	What local skills are available? Are features in line with skills? Are free and open source alternatives considered? Are the systems well documented?	Are local skills and knowledge being developed? Are stakeholders actively involved? What new ways of working are introduced? What will the impact be in terms of organizational change?
	Installation phase	
Is all the equipment well protected?	Has the system been tested with all stakeholders?	Are all stakeholders involved in training program? Is the added value made clear?
	Operation/maintenance phase	
Is local capacity sufficient? Are spare parts easily available?	Are software maintenance skills available?	Is a support organization in place? Is the support organization able to support all stakeholders (e. g. gender issues)

4.4 Clarification of terms: design, development, implementation, production and use

Appropriate ICT is based on a system development life cycle that incorporates the phases *definition*, *design*, *construction*, *installation*, and *operation and maintenance*. This thesis refers to the first three phases as *production*, and the later two phases as *use*. Installation has also technical elements that lie on the border between production and use, but to a large extent installation is concerned with user training and institutionalization, which belongs to the domain of use.

Outside the AICT framework, particularly in less technical literature, the term *design* is often used in a broader sense, e. g. including production activities that span from definition to construction phases. It is also a term to describe design theories, which will be applied in chapter 9 to describe the appropriate IS development methodology.

The term *development* is used frequently to describe the activities of the technical system creation; for example, Ågerfalk et al. (2005) chooses to define development broadly as "any software development lifecycle activity" (p. 2). In this sense, development includes production and use of information systems. Here, development follows the definition of Ågerfalk et al.

Implementation is a term that is used to describe a variety of different things, depending on the context. In technical literature, implementation means the transformation of a technical specification into source code (van Vliet, 2008). In social oriented information system research, implementation is more concerned with the effects on organizations than with the production of systems. This view considers implementation in a "human and social sense, so that the system is used frequently by organization members or that it is considered valuable for work activities or coordination" (Walsham, 1993, p. 210). In this thesis, the latter will be referred to as *organizational implementation*. In relation to projects, implementation means the execution of the project plan.

4.5 Evaluation of the Appropriate ICT framework

The evaluation of the Appropriate ICT framework is based on events taken place and experiences gained during the OPUS project, which was carried out between 2005 and 2009. The project has run through all stages of the software development life cycle.

However, only little evidence has been gathered concerning the operation and maintenance phase, since information system use only started very late in the OPUS project. Several Mozambican universities were part of the project, which provides a somewhat rich picture of project events. Since the Appropriate ICT framework was published after the official end of the OPUS project (van Reijswoud, 2009), it has not been applied during the OPUS project. The assessment of the Appropriate ICT framework is based on a reconstruction of project events. First, the usefulness of the Appropriate ICT framework is evaluated, including its tool usage and guiding questions. Questions of interest include:

- In which way is the Appropriate ICT framework useful during project lifetime?
- How would the use of the Appropriate ICT framework have improved the outcome of the OPUS project?
- Is the set of guiding questions complete? Are there suggestions for improvement?
- How can tool and method development be facilitated in order to make them reusable between ICT projects?

In the following, the project characteristics and events are outlined that are relevant for the analysis. Afterwards, the results of the assessment of the AICT framework are presented. First, the overall usefulness of the Appropriate ICT framework is reported. Then, the two elements of the framework are viewed separately: the guiding questions are analyzed for completeness, and guidelines are presented for appropriate development of tools and methods.

4.5.1 The OPUS project

The OPUS project was part of a larger project, containing several components, including *infrastructure*, *student information system*, *human resource development*, *e-learning* and *curriculum development*. Table 4.2 shows the relevant project components and their relationship with Appropriate ICT components. Hardware and software components were covered by the OPUS project. But in the area of organizational change the OPUS project did not cover all areas that are considered necessary by the AICT framework. The OPUS project did not sufficiently foresee issues like the organizational implementation. Installation only included technical steps, but user training was limited and there was no way for the integration of user feedback after the first experiences in a production environment.

Table 4.2: The OPUS project components in relation to AICT components

Project component	*AICT component*	*Equivalency*
Infrastructure	Hardware	Equivalent
Student information system	Software	Equivalent
Human resource development	Organizational change	The project component only covers a subset of the activities necessary to achieve successful organizational change

4.5.2 General observations of Appropriate ICT framework

The following arguments can be made in favor of the usefulness of the Appropriate ICT framework in the OPUS context:

- It assists in elaborating a complete picture of the project reality, by discovering potentially ignored aspects required for ICT project success.
- The guiding questions can be applied repeatedly throughout the project lifetime to react to changing project realities. Hence, they facilitate a dynamic unfolding of the project reality.
- Analysis through guiding questions can assist in proper tool selection, by validating if selected tools are likely to contribute to project goals, given the project reality.
- Appropriate ICT is a general framework guiding a wide variety of ICT interventions, based on the system development life cycle model. Therefore it can serve as a basis for more specialized methodologies, such as collaborative inter-organizational information system development.

4.5.3 Completeness of the guiding questions

The Appropriate ICT framework's guiding questions are the means of the Appropriate ICT framework to address relevant issues related to hardware, software and organizational change management. The questions were therefore answered to reflect project challenges. In a second step this picture was contrasted with the actual situation of the OPUS project to evaluate the usefulness and completeness of the picture drawn by the Appropriate ICT framework. The results of this comparison were indicative enough to suggest

some adaptations to the Appropriate ICT framework for the case of IS projects.

The guiding questions comprise questionnaires in three areas: hardware, software and change management. The completeness of the questionnaires is considered as follows. The hardware guiding questions gave a sufficiently complete picture of the project reality by revealing given challenges with Internet connectivity, environmental factors and staff competency. The software questionnaire covered many important aspects. However, the following should be considered and integrated into the software dimension of the guiding questions:

- In order to analyze the complexity of a project it is relevant to distinguish if existing software is used or software is developed as a part of the project. In the latter case the complexity of the project is increased and technical staff is needed to make system improvements – not only during the initial construction phase, but throughout the system's life span. In this case, the life cycle phases are typically carried out repeatedly.
- Software requires skills in three different areas: user skills, technical skills for system maintenance and software development skills. This distinction is not present in the questionnaire.

In the area of organizational change the questionnaire was able to spot shortcomings in the OPUS project. However, certain issues were missing in the questionnaire, including questions about:

- Who is the functional owner of the system, sometimes called *focal point*? (installation phase)
- How will user feedback be integrated into further system development? (operation phase)

4.5.4 AICT in relation to the overall software life cycle

Because the AICT framework is applicable to a wide range of ICT interventions, it lacks certain aspects specific to systems with a major software part. The basic system development life cycle, which runs through stages from definition to operation, is typically applied iteratively throughout the system's life span. But the characteristics of the cycles may change over time; early cycles may span longer time frames, focusing on initial development and introduction of the system into an organization, whereas later cycles may

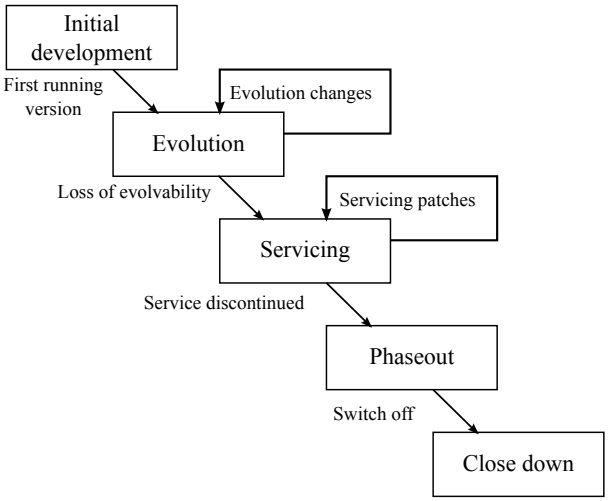

Figure 4.2: Staged model of the software life cycle (Rajlich and Bennett, 2000).

cover shorter periods of time and be oriented more towards small, sporadic improvements to the system. The extension to new contexts, the withdrawal of donors or the emergence of new donors are also factors that may change the focus towards the system and towards tool selection. For example, in many cases donors withdraw after the development of pilot systems. In other cases, donors come in to support the scaling of pilot systems.

It has been shown that software often evolves in stages. A staged model contrasts with the simplified view that all activities after initial delivery are considered simply as *software maintenance*. But this is a simplified view that does not distinguish between bug fixing and adding considerable functionality. In analogy to building a house, repainting a room or fixing a leak in the roof are called maintenance, but adding a wing to the office is usually not considered maintenance any more (van Vliet, 2008). In software development, if this distinction is made, one consequence is that software development is cyclic. Moreover, the evolution of software runs through several stages. The model in figure 4.2 is an alternative to the simplified software maintenance model and distinguishes five stages (Rajlich and Bennett, 2000):

Initial development: Construction of a first functioning system

Evolution: Extensions in functionality to meet user needs, possibly in major ways

Servicing: Focus is on bug fixing, and only small changes to the functionality of the system

Phaseout: Servicing activities stop, but the system is still used

Close down: The system ceases to be used, and possibly alternatives are put in place

Commercial software systems tend to progress along the five stages, that is, from later stages there is no turning back to earlier stages. For open source projects, however, it has been shown that software system development in some cases is able to move back to the evolutionary stage, even for projects that have passed this stage already (Capiluppi et al., 2007).

The staged life cycle model puts the basic system development life cycle into a broader context. Cycles running from definition to operation can be repeatedly executed in each of the stages. This is relevant in relation to the AICT framework, since different stages may call for different tools and methods, and possibly also different guiding questions.

4.5.5 Appropriate IS cyclic model

Figure 4.3 shows an adapted version of the AICT cyclic model, taking into account the observations made in relation to software centric information systems. The evolutionary characteristic is highlighted. Most of the phases identified in the staged life cycle model involve iterations: cycles are executed throughout the stages of initial development, evolution and servicing. The cycles consist of identifying requirements, changing the software accordingly, and evaluating it during operation. The first cycle is entered given a problem situation that is to be met with an IS project. During initial development and evolution, a lot of iterations may occur with varying degree of intensity. During servicing, the number and intensity of the iterations may decline. Finally, when the system is phased out, the iterations stop; no further changes are being done to the system.

4.5.6 Description of tools and methods

The Appropriate ICT framework in its current form does not provide guidelines for the description of tools and methods (referred to as tools in the following for the sake of simplicity), although a good description makes it easier to understand and reuse them in other projects. The analysis of tools used in the OPUS project suggests a common set of characteristics that are helpful for their description. The following is a possible set of tool description questions for the Appropriate ICT framework:

- What is the context for which the tool is appropriate?
- What is the problem to be solved by the tool and what is the goal to be accomplished?
- How does the tool relate to local culture, environment, organization, economy and/or

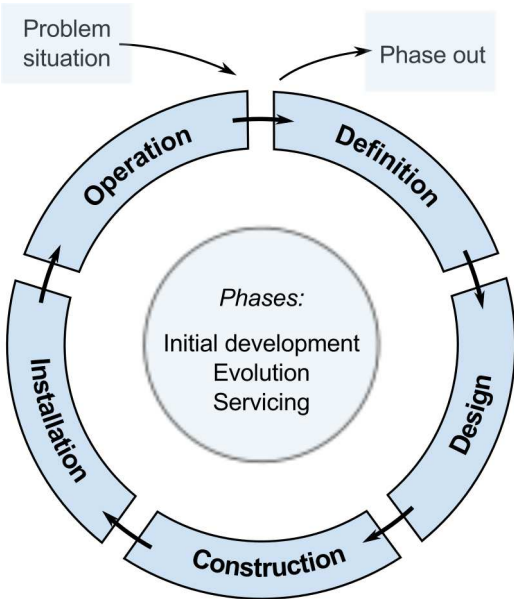

Figure 4.3: Appropriate IS cyclic model (adapted from Appropriate ICT model).

politics?

- Which stage(s) of the staged software life cycle does the tool apply to?
- Which step(s) of the system development life cycle (SDLC) does the tool apply to?
- What are the necessary preconditions, e. g. availability of resources?
- What potential difficulties can arise as a consequence of the application of the tool (risks)?
- What practical experience has been gained by applying the tool?

Table 4.3 shows an example how a tool can be described based on the given set of questions.

4.6 Chapter conclusions

Two sets of conclusions can be drawn in this chapter. First, the review of software development characteristics in relation to the ideas of appropriate technology allows for general conclusions concerning the design of information systems in the context of developing countries. From the evaluation of the Appropriate ICT framework, more specific conclusions can be drawn regarding the formulation of an information system development methodology that is appropriate to the context of developing countries.

Table 4.3: Example of an Appropriate ICT tool description

Tool:	Involvement of less experienced software developers during system development: E. g. report and scholarship module development on Mozambican side.
Context:	Globally distributed cooperation project where foreign software development experts needs to transfer knowledge to a local software development team.
Problem addressed:	Missing capacity of local software developers.
Description:	The tool is about giving parts of the system development to the local side – at the beginning small, but increasingly bigger parts with greater responsibility. This involves communication and knowledge sharing. The process may be initiated by a proper training of the relevant aspects of the system.
Stage:	Initial development, evolution
Phase:	Construction
Goal of tool use:	Local software developers shall be able to execute basic software development tasks for the application.
Potential difficulties:	- Local software developers need guidance. This is difficult if the competent contact persons are far away in a remote country. It can be facilitated by the presence of an external consultant who stays with the community of local software developers for a considerable amount of time. - Conditions for local software developers may not always be appropriate. They may have other duties besides their involvement in the project, so they need the commitment by their superiors, sufficient time and proper equipment. - The structure of the software system and its code base needs to allow that people in different locations around the world do not interfere with each other when working on the system. This can be facilitated by a modular structure and a central code repository.
Practical experience:	- The experience in the OPUS project is that local guidance and proper working conditions make a big difference. After a three-weeks developer workshop in the Netherlands one team of local software developers got the task to construct a scholarship module. Although the module was specified and designed, the local team did not go ahead with their assignment after they returned to Mozambique. They had other duties and missing commitment from their superiors concerning their activities within the OPUS project. - Another example was the development of the reporting module, which was successfully developed by another local team supervised by an external consultant. Subsequently, many other tasks have been done by the same local programmer under supervision and increasing communication with the team in the Netherlands via email and a mailing-list. In this case, local competency has been created, which stayed present at the university to further improve the information system for their own needs.

4.6.1 Appropriate technology lessons for IS development

Appropriate technology is relevant for information system development in the context of developing countries, because it focuses on problem solving with a strong involvement of the local community and the usefulness in local conditions. Rather than importing technology from outside, which is currently the main mode of technological innovation in developing countries (Alkhatib, Anis, and Noori, 2008), appropriate technology attempts to follow a route that requires considerably more effort: designing technological solutions that fit the local needs, given local constraints, in a way that builds capacity in a sustainable way, so that the technological solutions can be maintained by the local community or organization.

On the positive side, the activity of programming, an important building block of software development, requires little investment, and is labor-intensive. This corresponds to the appropriate technology philosophy. However, software development requires relatively high levels of knowledge, which is a challenge to the appropriate technology principle that local actors shall be able to develop and maintain local solutions. The required skills cannot be taught exclusively in training sessions, but experience plays a major role. This implicates a focus on an enabling environment where capacity building is nurtured over extended periods of time. Such an environment needs to find ways to make all actors part of the knowledge generation process.

4.6.2 Appropriate ICT as a basis for an IS development methodology

Although the analysis of the project reality based on guiding questions produces valuable insights that are useful for proper tool selection, the Appropriate ICT framework does not provide particular indications on tool selection. At least two approaches are possible to improve the mapping from project reality analysis to the prescription of appropriate action in order to achieve project goals.

One approach is a more detailed description of tools and methods. An example of this approach has been given in table 4.3. This is a low-level approach that can lead to an inventory of tools and artifacts. Individual tools may be subject to investigation, e. g. through case studies or action research. A high-level approach would be to derive a methodology that is more specialized to particular circumstances than the general Appropriate ICT framework; the AICT framework applies to a wide variety of ICT projects, but

for IS projects a more specific framework could be useful.

The two approaches do not contradict each other, but may strengthen each other. As identified in this chapter, important elements of an appropriate technology inspired methodology include community building within an actor network for collaborative system development, and the building of local capacity.

Both approaches will be followed up in subsequent chapters. For the lower level, tools and methods will be derived with aims such as setting up an open source structure (chapter 5), a framework for local innovation (chapter 6), and a support network (chapter 7). The high-level approach is used to seek a methodology for collaborative information system development with a diverse set of actors who bring different levels and areas of knowledge as well as different cultural mindsets to the table. The methodology is elaborated in chapter 9.

Chapter 5

Open source: How can complex IS be developed collaboratively?

The objective of this chapter is to show the opportunities, challenges and ways to approach information systems development in diverse actor networks with open source concepts. Possible building blocks for open source IS development are derived.

Free and open source software has gained momentum in both developed and developing countries. In developing countries it has been hailed as a chance to overcome specific problems in software development, such as dependencies on foreign software vendors, the limitations of the import of foreign developed software, relatively little existing technical skills and the high cost of software licenses (Ghosh, 2003; Rajani, 2003; Weerawarana and Weeratunge, 2004). Not least, the higher education environment is seen as an ideal ground for open source initiatives (Tong, 2004).

However, a significant participation of developing countries in open source initiatives so far has been mostly limited to the use of a few specific applications, such as the Linux operating system, the Mozilla application suite and the Apache web server. Like in other parts of the world, open source has a higher penetration on the server side than in the domain of end user applications (Wichmann, 2002). One possible reason is that open source traditionally favors technically interested *user-developers*, i. e. those users who possess the technical skills to modify source code and improve the software.

Thus, despite a high theoretical potential for open source in developing countries, it seems difficult to realize this potential practically, despite the observation that the ad-

vantages of open source software are getting confirmed by the organizations and individuals adopting it (van Reijswoud and de Jager, 2008). The observed advantages include reduction of costs, avoidance of vendor lock-in, unrestricted distribution of software and an increased understanding of computing at all levels involved.

Merely using open source software without further local adaptation already offers certain benefits, like the absence of license fees. But by becoming active software producers, even more opportunities arise, either by improving existing applications or developing new ones.

One frequently mentioned problematic issue is the scarcity of locally available skills. In order to take advantage of the possibilities of collaborative open source software development projects, all parts of the network need to have certain capacities (Braa et al., 2007). Hence, a capacity building element needs to accompany any initiative to form an open source network.

Another key element is the utility, or relevance, of the solution that is being developed (Heeks, 2005b). This aspect requires ongoing attention throughout the system life cycle. Activities at all stages, from definition to operation, have influence on the practical value that is being created for its users.

Although many universities in developing countries suffer from difficulties related to the management of their academic data, up to now there are hardly any open source products available for student information management in developing countries in general. For the context in Mozambique, no suitable open source options were encountered. This lack can be attributed to the complexities involved with software use and development, stakeholder coordination and required domain knowledge.

Similar to other domains like financial management, using an information system to support academic student registration is a common approach for universities in developed countries. More and more, also in developing countries the use of effective information systems is seen as a *conditio sine qua non* for success according to national and international standards. In this situation where the demand is not met by neither affordable commercial nor existing open source products, the collaborative production of an information system is a potential option, and open source licensing provides a possible contractual framework for the partners, including yet unknown future partners. The OPUS project, a collaborative open source project to develop a student information system, provides a rare opportunity to study and learn lessons for other open source projects both within and outside the realm

of academic institutions.

5.1 Initial methodological considerations

In the context of actor networks with varying degrees of capacity, as is often the case in north-south projects, a principal objective is to establish the conditions that the information system can be sustained after the withdrawal of more experienced nodes, or after project funds have dried up. As a conclusion from previous chapters and as a guide for further methodology development, elements of the overall goal of sustainability include:

- To support system operation at a low level, i. e. the basic running of the system
- To make the system sufficiently configurable and adaptable to the contexts of implementing universities
- To build capacity for maintenance and further development
- To stimulate continuous improvements and adaptation based on emerging user needs, in order to maintain the system's relevance
- To enable inter-organizational community building by currently involved and potential future partners
- To provide external support to users and organizations, i. e. someone to get in contact with in case of difficulties
- To foster local ownership

The theoretical considerations in this chapter will take into account issues including software licensing in general and open source licensing in particular, stakeholders and their interests, governance of open source projects, economic and motivational factors for participation in OSS, and system design aspects concerning modularization and internationalization.

Several levels are addressed: first, the level of a particular user organization (e. g. a university), second, the regional community that is formed through the common use and interest in the same software (e. g. the Mozambican community of universities, support organizations and Ministry of Education), and third the global level that includes international partners (e. g. the development cooperation partners) and potential future participating universities that may get interested in the system once it is made publicly available.

5.2 Free and open source software

Based on the initial observations made in the previous section, theoretical concepts are discussed in this section, which will be used to elaborate open source building blocks in the next section.

5.2.1 Clarification of terms: Free, libre, open source

Free software and *open source software* have different underlying motivations, and not all software licenses that are recognized as open source licenses are recognized as free software licenses and vice versa. The word *free* in *free software* is concerned with liberty, not with price, that is, "free as in free speech, not as in free beer". A program is considered free software if users are enabled (a) to run it, (b) to distribute it, (c) to change it, and (d) to distribute the changed version of the program. Access to the source code is a precondition for these freedoms (Stallman, 1996). Free software is inspired by ethical and social values, and the Free Software Foundation (FSF) campaigns for these stated freedoms. Because of the ambiguity of the term *free* in the English language, some have proposed to use the term *libre*, which in the Spanish language specifically designates the intended meaning of *free* as described above.

Like free software, open source software is also based on the free access to source code. It differs from free software in that it puts emphasis on the practical benefits of its licensing practices rather than on moral rightness. It is a development approach and detached from the ethical values expressed by free software. Open source software development grew out of the free software movement with the objective to attract more corporate contributions.

Despite ideological differences, many developers feel little impact caused by the applied license on their actual software development activities in their communities (Eilhard, 2009). Furthermore, nearly all open source software is free software (Stallman, 2007). The term *open source* is now generally used by scholars to refer to both free or open source software (von Hippel, 2003), and this is also the chosen term in this text.

On a further note, it shall be emphasized that neither free software nor open source software are to be confused with *freeware*, which refers to the free availability of the executable program, but not necessarily to the source code. Freeware plays no role in the further theoretical considerations in this thesis.

5.2.2 Opportunities, challenges and culture of open source

Commonly stated advantages of open source in relation to closed source software development include robustness of the code, better security, flexibility to the user (e. g. avoiding vendor lock-in) and support from a community (Krishnamurthy, 2003). Braa et al. (2007) outline opportunities and challenges in open source software development within a south-south-north network of participants: "Sharing of resources is one of the great promises of Free and Open Source Software (FOSS) approaches to software development – but it also puts demands on the local capacity in all parts of the network, ranging from software development to adapting the use context – capacity to both meet local needs and contribute to global development at the same time."

Open source, as a form of open innovation, has the potential to overcome limitations inherent with the proprietary model in which source code is kept secret by the organization that is owning and producing a particular software program. The potential of open innovation stems from the opportunity of tapping a broad range of knowledge sources to nurture innovation, such as customers, rivals, universities and organizations in unrelated industries. Open innovation also includes the creative management and use of the artifacts created in the innovation process. One of the principal challenges for the open innovation model, and particularly for open source, is to motivate contributions from external sources of innovation (West and Gallagher, 2006).

In comparison to commercial software, open source software tends to put less emphasis on documentation, support, user interfaces and backward compatibility. The greatest diffusion of open source projects appears to be in settings with sophisticated end users such as Apache servers. Advanced users and user-developers are better able to cope with the listed shortcomings (Lerner and Tirole, 2002). The focus on user-developers is also reflected in the historical development of open source. In the 1970s open source was limited to small projects such as demos. In the 1980s development tools and Internet tools emerged. In the 1990s the attention shifted towards operating systems, e. g. Linux. Over time, increasingly difficult problems were addressed by open source. However, programs were developed mostly for use by developers. Only in the 2000s open source entered the realm of applications for less technically advanced users. One of the early examples is *Gimp*, a powerful image editor application similar to *Photoshop* (Raymond, 1998).

Whereas commercial software production follows hierarchical structures, open

source software production does not have an obvious method to resolve disputes. Hence, a particular challenge associated with open source software is *forking*, the development of different – eventually competing – versions of the same software. A fork may happen in case of conflicting opinions of developers concerning the future of the open source project. This indicates the importance of sound governance of open source projects (Kogut and Metiu, 2001).

Contributors to open source software have a culture of "maximizing reputation incentives", of ensuring that peer credit goes where it is due and does not go where it is not due. Among open source contributors there exist three taboos to avoid damage to their reputation: (1) forking of projects, (2) distributing rogue patches and (3) "surreptitiously filing someone's name off a project". The latter is considered one of the "ultimate crimes" (Raymond, 1998).

5.2.3 Governance

Leadership in open source projects is an important determent of project success. A common feature of many open source project leaders is that they are programmers who made important contributions early in the project's development, and moved on to broader project management tasks. The leader in an open source setting often has no formal authority over programmers, but his recommendations tend to be followed by the majority of programmers in the project. Leadership has an important role to play in accepting or rejecting modifications to the code in order to keep a required level of quality, especially because of the absence of liability as would be the case if sold by a commercial software firm. A key to successful leadership is the trust by the programmers (Lerner and Tirole, 2002).

The benefits of the open source production method are limited by the quality of the coordination process, by the level of redundancy of development and by versioning problems (Kala, 2008). Bad governance can result in delays, redundant production or the forking of the project. Effective governance prevents free-riding, coordinates the software development activities and ensures the quality of the outcome. There are three typical kinds of leadership in open source projects: charismatic leader, voting committee and rotating leadership (Eilhard, 2009).

Although the term *governance* has been used repeatedly in the context of open

source software, there is no common definition, possibly because of the multiple dimensions of governance that are being discussed in the literature. One rare attempt to define OSS governance is the following (Markus, 2007, p. 152):

> OSS governance can be defined as the means of achieving the direction, control, and coordination of wholly or partially autonomous individuals and organizations on behalf of an OSS development project to which they jointly contribute.

This definition is open to include four dimensions of OSS governance: (1) structures and processes; (2) informal, formal, and encoded rules; (3) externally applied as well as internalized rules; and (4) mechanisms of both trust and verification/control. Different projects use different variations of these dimensions in their leadership style. For example, community initiated projects can be expected to have different governance characteristics than sponsored open source projects.

The purpose of open source software project governance can be distinguished into three different positions (Markus, 2007):

Solving collective action dilemmas: This is about the social challenge of motivating participation to an open source project and providing incentives for individuals and organizations for participation. Examples for such incentives include governance mechanisms such as OSS licenses and not-for-profit foundations.

Solving development coordination problems: This deals with the routine challenges of coordinating interdependence in getting open source development work done. One example is how individual contributions can result in high quality open source products, which can be achieved with strategies such as peer review of source code.

Creating a climate for contributors: This view contrasts with the other two views in that an open source government approach is seen as an end in itself rather than solving social or routine challenges. It intends to create a good organizational climate that increases the individuals' motivation to work. The reasoning is that organizational climate can influence individuals' motivation independent of benefits such as monetary incentives and recognition. Therefore, a good open source project climate based on democratic governance may be more effective at motivating contributions than formal rewards and private benefits.

The three different purposes of government are not mutually exclusive, that is, in a single project, open source governance may be able to address all of them. However, it is unlikely that a single governance mechanism, for example the application of an open source

license, can do so effectively. Other techniques need to be put in place additionally. For example, to overcome reservations of contributors, Shah (2006) suggests several governance mechanisms how a leader can interact with the community: (1) involve contributors in the decision-making process, (2) use a restrictive open source license that effectively prevents corporate code hijacking, or (3) hire renowned open source developers to show the commitment to the open source idea. These recommendations are particularly relevant in the case of company sponsored projects that seek volunteer contributions while at the same time limiting the rights granted to the volunteers; in the extreme case the source code stays in the possession of the company to make profits by selling commercial licenses. Developers generally are suspicious to corporate run projects, because contributions can be hijacked for company profits without developers getting recognition for their contribution.

5.2.4 Software licensing

In many countries source code is protected by copyright and intellectual property regulations. As soon as source code is created, it is automatically protected by copyright. Therefore, source code cannot be used legally by others without granting appropriate usage rights. Software licenses define the sort of rights that are given to other users. The spectrum of licensing options ranges from public domain to open source licenses to commercial licenses.

When an author puts his source code into the public domain, others are allowed to use it in any way they like. With commercial licenses on the contrary, the allowed software use is accurately specified and limited, for example as the right to install and run a single copy. The user might also be limited by a time-based or seat-based scheme. Open source licensing can be placed in the middle of the range and consists itself of a spectrum of licenses. Open source licenses range from academic and permissive to restrictive licenses. See table 5.1 for an overview of important types of licenses.

Academic and permissive licenses give a lot of freedom to the use of the source code. Academic licenses are typically short because of only few usage restrictions. They primarily keep names and copyright notices intact. Permissive licenses are slightly more complex. They protect the authors from being legally challenged by users, for example concerning trademark names and logos. Academic and permissive licenses allow products derived from open source code to be redistributed in a closed source fashion. In practice, this is often not a problem, because it is cheaper and easier to integrate improvements

Table 5.1: Recommended open source software licenses, ordered by improving restrictiveness (Lindberg, 2008)

Type of license	*Characteristics*	*Recommended licenses*
Academic licenses	Preserve copyright notes	2-clause BSD license
Permissive licenses	Legal protection	Apache License version 2.0
Partially closable licenses	Require sharing of parts	Mozilla Public License Lesser GPL (LGPL) version 2 or 3
Reciprocal licenses (*viral* licenses)	Share entire application	GNU GPL version 2 or 3 Open Software License (OSL)

with the main line of the product than closed-source redistribution. Examples for academic licenses are BSD and MIT licenses. An example of a permissive license is the Apache V2.0 license.

Partially closable licenses allow certain parts of the code of an application to be distributed freely, e. g. commercially, while other parts of the code are required to stay open source. Because of this separation, partially closable licenses usually show up in two areas: libraries and extensible applications. For example, libraries licensed under the LGPL can be redistributed in proprietary applications in unmodified form, but any changes to the source code of LGPLed code will have to be made available to the public. Applications that include LGPLed libraries are not required to apply open source licensing to the whole application. On the other hand, extensible applications have an open source core and allow proprietary extensions. An example of an extensible application is the popular Eclipse software development environment, which is licensed under a partially closeable license and permits open and closed source extensions.

Reciprocal licenses, such as the GPL, require that binary distributions include the full source code of the application. They are sometimes called *viral* licenses because of the requirement that applications need to apply open source licensing on the whole product if any part of the source code is licensed under a reciprocal license.

Due to the different restrictions of different kinds of licenses, the compatibility of licenses is limited. If a given software system is licensed under the LGPL it may be redistributed in other software under the GPL, but not the other way around. Another aspect of license compatibility is related to the big number of different licenses. Because of com-

patibility issues, it may discourage possible contributors if less popular licenses are applied. To minimize compatibility issues, Lindberg (2008) recommends to limit the choice of open source licenses to a small set that should be sufficient for the majority of circumstances (see table 5.1). An analysis of projects hosted on Sourceforge made the following observations about license choice (Lerner and Tirole, 2005):

- Restrictive licenses are more common in projects geared towards end users, like desktop applications, in languages other than English, and designed for a non-commercial user environment or operating system.
- Community contributions are greater when restrictive licenses are employed.
- Software that is made available under nonrestrictive licenses is particularly prone to hijacking by commercial software vendors, meaning that somebody may add some proprietary code to the software and take the whole private.
- A more restrictive license choice reduces the opportunity for earning money on complementary products.

License choice plays a role as a stimulus for community building and economic development. A good license choice allows and motivates local involvement and ownership and continuous development and adaptation. A proper licensing strategy is therefore a prerequisite for sustainability.

5.2.5 Community initiated vs. spinout projects

Open source projects are created in different ways: either as *community initiated* projects or by releasing code that has been developed by a sponsor internally. The latter is called a *spinout* project. Spinout projects enter the open source domain with a software system that typically is already usable, whereas community initiated projects start from scratch and have yet to overcome the hurdle to produce a useful artifact. Community initiated and spinout projects have different reasons for initiation, key issues, motivation for contribution and control mechanisms. West and O'Mahony (2005) have summarized these issues, as shown in table 5.2.

The way open source projects are created influences the stability of the community and the degree of local ownership. Community initiated projects tend to create a stronger degree of ownership because of early contributor participation, but such projects face the hurdle to reach a stable product. Spinout projects, on the other hand, have to find a balance

Table 5.2: Community initiated versus spinout projects (West and O'Mahony, 2005)

	Reason for initiation	*Key issues*	*Contributor motivation*	*Control*
Community initiated	Solve a problem Create a *free software* alternative to proprietary solution	Garnering resources Building healthy community, attracting talented developers Distributing software Gaining 'mindshare' with minimal marketing	To make software happen To gain fulfillment To build and learn new skills To solve personal and professional problems	Democratic, transparent, usually meritocratic Some leadership and stratification
Spinout	Achieve greater adoption Get development help on areas that are of low priority for the firm (e. g. special dialects)	Gaining legitimacy Building healthy community, attracting talented contributors Resolving ambiguity about control and ownership	To complete areas that are of high priority for contributors To gain visibility by prospective employers To influence sponsor's alignment with complementary projects	Varies but sponsor usually retains direct or indirect control

between sponsor control and attracting contributors.

5.2.6 Economic sustainability

The philosophies between free software and the free market are quite different. Open source software is related to cooperation, inclusion, sharing and openness, while the market is related to competition and self-interest (Chege, 2008). The recent popularity of licenses that are more liberal than the restrictive GNU GPL license shows pragmatism to combine the advantages of open source with for-profit activities. Historically the greatest diffusion of open source has been in settings where end users are sophisticated, e. g. system administrators. These advanced *user-developers* tolerated a lack of detailed documentation or easy-to-understand user interfaces. With increased business focus open source software enters segments that it traditionally served poorly (Lerner and Tirole, 2002).

Richard Stallman, the president of the Free Software Foundation (FSF), proclaimed: "We encourage people who redistribute free software to charge as much as they wish or can. [...] Distributing free software is an opportunity to raise funds for develop-

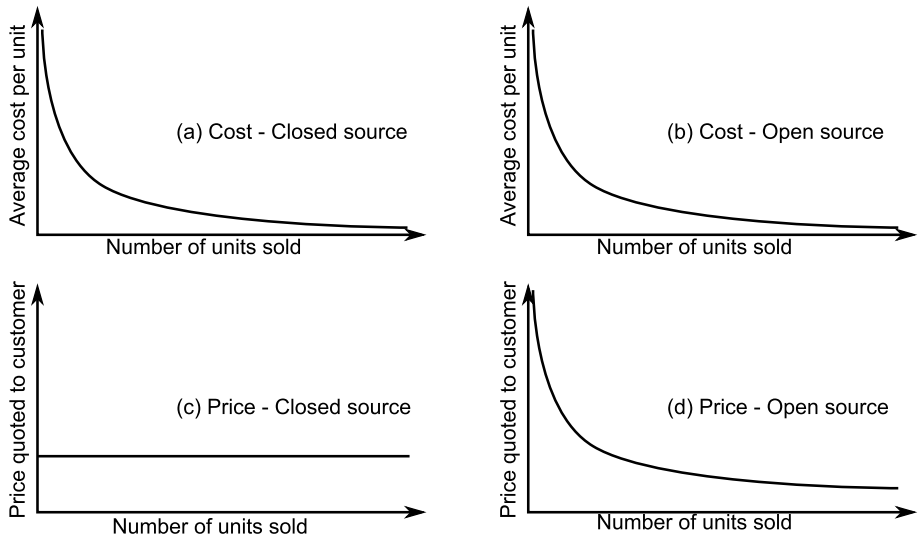

Figure 5.1: Cost and price of closed source and open source software

ment. Don't waste it!" (Stallman, 2002, p. 65). However, although this statement marks no objection to earn money with software, the GPL puts limitations on how to use the software licensed with it. Therefore, business with GPLed software is hardly done by selling the source code unless the software developers are contracted by someone to carry out particular software development tasks. Profit is rather made by third parties who sell services around the software, like training, support, distribution, hosting and consulting. GPLed software is typically geared towards sophisticated users. With more permissive licenses (see table 5.1), products derived from open source software can be redistributed in closed source fashion. This creates possibilities for income (Krishnamurthy, 2003).

The total price that customers have to pay for IT products can be broken down into three components, namely hardware, software and services. For customers of open source based applications, the total price is usually lower than for closed source software once a certain minimum of units has been sold. Production cost and selling price of closed source and open source software are illustrated in figure 5.1. The price of obtaining open source software is closely related to its production cost. The development of the first copy of a software product is typically most expensive. For subsequent sales, the costs tend to decrease. Closed source software may have a similar cost structure, but the price stays stable. This is an advantage for early customers and a disadvantage for later customers. The total revenues from sales may be similar from open source and closed source software.

But vendors of open source software have a higher chance of receiving help from customers, system integrators and others during product development than closed source software companies, which is a potential advantage of the open source production method. (Riehle, 2007).

The most common open source businesses model is one of two kinds of *open source service companies*; one provides first level-support and installation services, the other provides second-level support, training and development services. Clients of first-level support companies are typically users who want to integrate an open source product into their IT operations. Clients of second-level support providers need to get trained on the product or need a fix to a technical problem that they cannot handle themselves. The strength of services businesses are related to (a) setting up and executing specific services, (b) application of expert domain knowledge and unique intellectual property, and (c) having the right people (Riehle, 2007).

5.2.7 Favorable project characteristics, modularity

Favorable characteristics for open source production include project modularity, existence of fun challenges and credible leadership that provides vision and keeps the project together, i. e. prevents forks. At the start of an open source project, there shall be enough code present so that a community can react as well as get convinced that the project has merit.

The aspect of modularity, relevant for the management of software projects in general, is considered essential for the viability of open source software. Baldwin and Clark (2006) argue that the architecture, particularly modularity, is a critical factor that lies at the heart of open source development, because modularity allows module designs to be changed and improved over time without undercutting the functionality of the system as a whole. Hence, a modular structure facilitates decentralized development by independent groups of developers. Kogut and Metiu (2001) go as far as to declare software without a modular structure to be inappropriate for open source development.

The decomposition of systems into modules is a basic practice in software engineering already for decades. Prerequisites for modular programming are (1) that modules can be written with little knowledge of code in other modules, and (2) that a selected set of modules can be assembled to form a functioning system. Modules shall not simply be con-

sidered as subprograms, but to assume a certain responsibility. A guiding principle for the decomposition into modules is *information hiding*, the conglomeration of related information that rarely needs to cross module boundaries. The expected benefits of modularization comprise the following (Parnas, 1972):

Managerial: Shortening of development time because separate groups work on modules without the need for much communication.

Product flexibility: Possibility to make drastic changes to one module without having to change other modules.

Comprehensibility: Easier understanding of the overall system because it can be studied one module at a time.

5.2.8 Open source in the developing country context

There is considerable interest about the use of open source software in developing countries, but opinions vary if and how it can be useful. According to Heeks (2005a) there is a lack of strong evidence of FOSS benefits. He questions it to be a *blind alley* for developing countries. A more optimistic view is expressed by Lerner and Tirole (2002, p. 198): "Users in less developed countries undoubtedly benefit from access to free software". Walsham and Sahay (2006) consider open source software a relevant technology in the context of developing countries and emphasize the technological details of open source licensing agreements. Given such a variety of opinions, this section looks at open source software in developing countries, in order to verify if the benefits outweigh the con's.

On the governmental level, developing countries have begun to embrace open source software motivated by (a) a desire for independence, (b) a drive for security and autonomy, and (c) new intellectual property rights enforcement and productivity (S. Weber, 2004). Countries around the world try to minimize their reliance on single suppliers who may not be focused on the country's interests, and to avoid opportunism by suppliers because of vendor lock-in. Open source represents a possible route for more local participation in software development, thereby contributing to a local software industry, keeping expenditures within the region and effectively strengthening local independence.

In an attempt to highlight problems with proprietary software, Peruvian Congressman Edgar Villanueva in an open letter to Microsoft Peru (Villanueva, 2002) objected to the monopoly of data encoding and processing by a single provider, and argued that

the usability and maintenance of software should not depend on the goodwill of suppliers or monopoly conditions imposed by them. Other developing countries have also expressed grievances with proprietary software, pointing to the little influence they have as small customers on how software evolves. Open source is expected to provide more flexibility and to allow more autonomous input, fostering local ownership. With open source, an indigenous industry can potentially participate in both identifying and meeting software development needs.

With increasing attempts to fight software piracy throughout the world, including in developing countries, open source represents a strategy to comply with intellectual property regimes. But there is a deeper issue involved. The degree to which a software tool can be utilized and expanded is limited only by the knowledge and innovative energy of the users, not by exclusionary property rights, prices and the power of corporations. Free access to source code not simply costs less but enables learning by doing, which creates demand and enables the development of applications that fit specific indigenous needs.

Ghosh (2004) proposes three main reasons for using open source software in developing countries: (a) cost, i. e. total cost of ownership, (b) performance, flexibility, localization, and (c) skills development. Open source software helps localization, in contrast to proprietary vendors, who do not care particularly about local issues, which consequently tend to be ignored. Open source software makes local adaptations possible: “Many FLOSS developers may have absolutely no interest in software usability for Xhosa speakers. But FLOSS developers allow and encourage those with locally relevant motives to adapt their software” (Ghosh, 2004, p. 22).

Open source software stimulates the development of local skills; not only the skills to use free software applications, but also skills like programming and teamwork. These skills are learned by participating in the open source community. Instead of solely using a software system, it allows active creative work on the software artifact. This is important since skill development requires access to the ability to create. Ghosh deducts that this low entry barrier to creativity enables a technology transfer mechanism from those in the open source community who have specific knowledge to those who do not yet. Hence, if open source is applied to north-south cooperation projects, technology transfer can be integrated into such projects.

5.2.9 Open source and technology transfer

Up to now, technology transfer between developed and developing countries is to a large extent limited to the direct import of technology, e. g. the purchase of foreign equipment. While this can provide quick operative results, little knowledge is transferred that would enable local control and the independent production and customization of technology. More generally, typical limitations of technology transfer to developing countries are (Alkhatib, Anis, and Noori, 2008):

Asymmetric information: The knowledge holder does not reveal information without incentives, and knowledge receivers cannot identify the value of information before buying it.

Market power: Technology owners are interested in reping profits and in covering the costs of innovation processes.

Limited movement of people: In developing countries incentives are typically too weak to support free movement of people and their knowledge.

Intellectual property rights: Costs of licenses and other fees.

This characterization might suggest that technology transfer has a uni-dimensional character, in which those in the industrialized countries are the sources and those in developing countries the recipients of information. But effective technology transfer is not a one-way information flow, but a two-way communication process. Transfer occurs through the communication of information. One should think of participants in the technology transfer process rather than sources and receivers. Each party involved in the technology transfer process may have a different perception of the technology. These differences can only be worked out through two-way communication (Rogers, 2002).

Open source emphasizes direct communication. Local innovation is not limited by intellectual property restrictions. In turn, the viability of local innovation reduces the brain drain. Open source can be used as a tool to make the step from exclusively foreign system development to the involvement of local developers in developing countries. It allows local learning and better coordination between the two sides in cooperation projects, because code is shared and local developers can take responsibility according to their current level of knowledge and steadily increase their experience. Foreign developers can increase their understanding of the use context through continuous communication within open source projects.

5.3 Towards best practice open source development

Based on theoretical considerations of appropriate technology and open source software, and on experiences in the OPUS project, this section formulates a model with building blocks for open source based cooperation projects.

The release of source code under an open source license provides access, but this is not enough to bridge the digital divide. The envisioned goal is to achieve effective use, which requires methods and tools that enable local participants to become active producers. Therefore, possible building blocks for cooperative open source projects need to facilitate local learning and ownership.

One of the requirements of the model is to handle the actual reality of existing local capacity, to improve local capacity over time to the required levels, and to substitute the missing capacity while it is not yet available locally. The given level of local capacity may vary in different local user organizations. It may also vary over time, for example due to staff fluctuations. Therefore it is desired that local installations are not jeopardized because of the reality of local capacity. The model presented in the following tries to achieve this by making the software development process a joint effort between participants of different backgrounds and locations, and by facilitating local learning during the process.

5.3.1 Model overview

The conceptual model (see figure 5.2) is based on the distinction between the global software core and localized versions of the software. The core software is the common building ground for all user organizations. It resembles the functionality that is considered useful for most instances of the software. Localized versions of the software are derived from the core through local adaptation, translation and enhancement.

The model's focus is on local users and on local capacity building. In order to realize the intended benefits in these areas, proper communication and coordination skills need to be established. Here, the development cooperation partner comes into play to cooperatively set up the structures, to encourage proper communication and to build the basic skills. An important element in this process is the installation of the system in a way that the system is considered valuable by its users. The installation effort helps to discover practical difficulties and to find ways to fit the technology with local needs. The particular relevance of practical organizational implementation for local learning was found by Braa

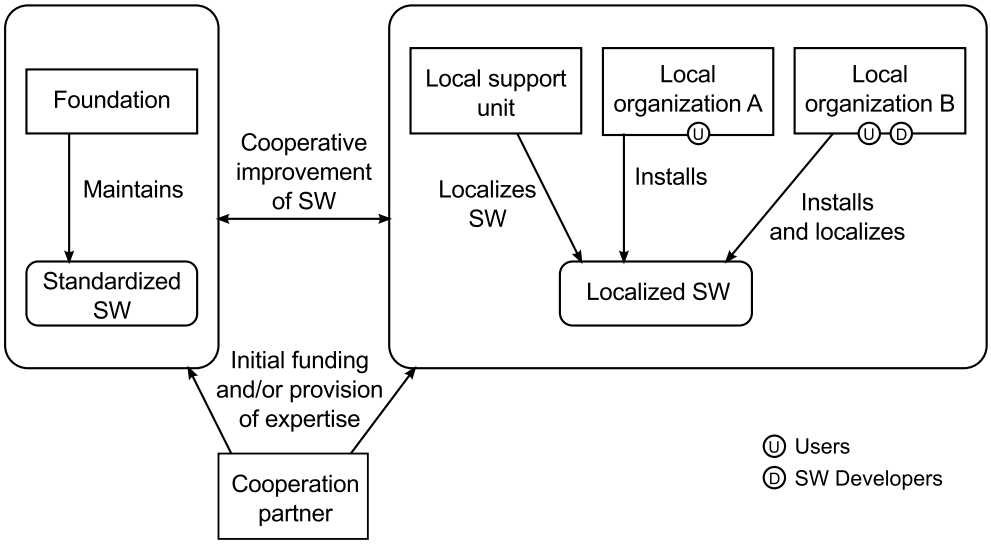

Figure 5.2: Conceptual model for open source software development

et al. (2007, p. 8) to be "arguably the most effective learning mechanism" in the context of setting up health information systems in several developing countries.

Open source offers the possibility for user centered innovation. Local software development is closer to the end users than software that is produced abroad and then imported. End users of application software can rarely be expected to be technically experienced *user-developers* themselves, especially in developing countries. However, certain closeness between developers and users is considered essential to streamline needs with software development activities and thereby facilitate the utility of the resulting software.

5.3.2 Localization versus standardization

Both standardization and localization are necessary. A balance needs to be sought between sensitiveness to local context and standardization across contexts. Standardization is necessary to control the potential chaos of excessive differentiation based on site-specific requirements. On the other hand, local variations, *workarounds* and tinkering are inevitable to deal with the shortcomings of standardization. It is important to notice that localizations are an important element in improving the standardized solution. Standardization and localizations form an ongoing negotiation towards improved standardized solutions (Rolland and Monteiro, 2002). An optimal balance between standardization and localization minimizes the cost of IS development.

One area to be balanced is the accuracy of information; the more data is required to be entered into the information system by its users, the higher the costs of collecting, registering, managing, and reusing such information. Different implementing organizations may demand different sets of mandatory data.

Such local variations may not always be catered for by the available configuration options of a standardized solution. Hence, sometimes *special cases* are needed. The decision which cases are considered special is important, because special cases tend to require additional work. For example, realizing special cases may necessitate an extensible, modular system architecture (Rolland and Monteiro, 2002).

As figure 5.2 indicates, there is supposed to be one standardized core system, and many local variations. The core system is managed by a dedicated legal entity. Local organizations will first evaluate the system and eventually make a decision to introduce it into their organization. They need to go through a process that consists of such activities as technical installation, configuring organization specific properties of the system, importing existing data and aligning internal processes. There may be certain requirements that are not covered by the given core system, and local enhancements will be desirable or necessary. In this case the user organization may use internal capacity or contract a third party, i. e. a support service provider, to develop the required enhancements.

When a user organization enhances the functionality of the system, for example by developing an additional module, then this enhancement can potentially be integrated into the core of the system, if it is of interest to other users. Vice versa, when the core software is improved, it can be used to upgrade local installations. Therefore, the system can cooperatively be improved by the local organizations and the maintainers of the core system. To find the boundary between core functionality and localized functionality is a challenge, though, and requires properly skilled staff.

A good balance between core and localization furthermore minimizes the tendency to fork. In some cases, for example due to time pressure during the installation phase, local developers may be inclined to change source code belonging to the core system to suit own needs. But if these alterations are not accepted to be incorporated in the global core then effectively a fork is created, which limits the update compatibility with the global core. Therefore, communication and proper care are important for both standardization and localization efforts.

The existence of an organization managing the global software core provides the

basis for collaborative development and technology transfer. It is the means to avoid local sides being left alone after the end of development cooperation projects. One could argue that the creation of a central organization in fact creates dependencies between user organizations in the developing world and the foreign central entity. But evidence shows that the gaps between required and existing capacity are often too big in developing countries to be bridged in the short term (Heeks, 2002b), making a longer term approach necessary. Until local capacities have reached the required levels, dependence on the central maintenance unit in fact exists. To counteract this phenomenon, a local support unit is established to support developing country partners and to facilitate local capacity building and local ownership.

5.3.3 Financial sustainability

The issue of generating funds to sustain continued evolution of the software system is related to the roles of the different stakeholders. The existence of the central unit is motivated by the fact that certain skills are locally unavailable. Local support units are included in the model to provide services to local organizations that (wish to) use the system. The needs that arise from users and managers in local organizations shall be served by multiple levels of support, which comprise (1) organization-internal support, (2) a local support unit and (3) the central unit. The local support unit may provide first level support, such as IS installation. The central unit may provide second level support, including the training of first level support unit staff, and the development of new functionality as required. While the goal is to solve support issues as much as possible locally, it is likely that the central unit will be needed on a regular basis. This collaboration forms the basis for financial compensation.

5.3.4 Building blocks

Table 5.3 shows an overview of building blocks for open source based development cooperation projects. These tools and processes are complementary to general open source techniques such as the use of source code repositories, issue trackers, and the culture of contributor recognition. The list of building blocks shall be understood as options, conforming to the statement of Darrow and Saxenian (n.d.) that the appropriate technology worker needs options, not a prescribed package of technology. It also conforms to the Appropriate

Table 5.3: Building blocks for open source information system development

Definition phase	
Local needs orientation	Ensure relevance and sufficient local interest as a prerequisite for long-term sustainability
Agreements	High-level agreements with local organizations to increase the likely involvement of local software developers and project leaders
Design phase	
Spinout source code	Provide a working product to the open source community
License	Balance community trust and economic opportunities
Construction phase	
North-south-south development model	Increasingly involve and give responsibility to local participants
Modular architecture	Allow different teams to work on independent parts
Build on open source components	Licensing of components may have influence on licensing of the entire application
Installation phase	
Development focus at installation	Reserve resources for development during installation at user organizations in order to deal with upcoming issues
'Concentrated installation package'	A recommended set of activities and deliverables for the practical installation of the system
Operation phase	
Local support structure	Multilevel support with emphasis on solving problems close to their source

ICT framework, with its invitation for the development of further tools along the system development life cycle (van Reijswoud, 2009).

Local needs orientation

Projects may face severe difficulties to be sustained if the reason for the project initiation is not firmly rooted in the needs of local participants. It is necessary to thoroughly understand the local situation so that projects are not abused for hidden agendas, for example to get access to project funded infrastructure without real interest in the primary project goals.

Agreements

Practical creative work is an effective method for learning. Therefore local software developers and domain experts shall get involved as early as possible in practical problem solving. Local involvement may be inhibited because of a frequently observed attitude towards development cooperation projects: Because of a tendency to consider the project partner from the north as active producers and givers, many see the partner from the south as passive recipients. This view often prevents active engagement by all participants, and promises of involvement by local organizations are often not kept. This vicious cycle needs to be avoided to give a chance to technology transfer based on two-way communication.

Before the initiation of information system development activities, agreements backed by high level management may be formalized that give an indication that the engagement in local learning is realistic during the development cooperation project lifetime. More specifically, agreements concerning local developers and domain experts are advisable, to make their availability for project activities more likely. This also helps to prevent investment in training of developers who would later not be available to apply the newly acquired knowledge.

Spinout open source

Under the spin-out model (see table 5.2, p. 141), first a working version of the software is developed before the source code is published under an open source license. During the initial development the focus is to start a community of software developers in the cooperation project with participants from all organizations, who have different levels of experience. When the software system has reached a certain level of maturity and usefulness, spinning out the source code can lead (a) to greater adoption of the software by organizations that have initially not been involved in the project, and (b) to a more stable community. A further possible positive effect of applying source code under an open source license is to help resolve ambiguity about control and ownership. Control is typically retained to a certain degree with a central entity. Courant and Griffiths (2006) call such centrally controlled projects *directed open source* projects.

License choice

Although the particular license choice depends on the context, some indications can be given. In many cases the specific license selection process needs to take into account community interests as well as commercial interests. A careful trade-off between the two is necessary in order to indicate proper licensing options. Community interests include the attractiveness of the software and the adaptability to local needs. Commercial interests have to be satisfied regarding financial sustainability, e. g. to ensure the funds for continuous development.

If funds need to be earned to keep up further software development activities, then a *partially closable license* may be an appropriate choice to combine advantages of open source collaboration with commercial interests. This possibility is particularly relevant with a modular system architecture. If the business focus lies on services, then a reciprocal license like the GPL may be the right choice. For recommended licenses see table 5.1 on p. 139.

North-south-south development model

The development model tries to combine two goals: The provision of a high quality software, and local capacity development.

- In the system development phases design and construction the core lead group of experts from the north builds the platform or kernel of the application with gradual introduction of developers from the developing country, for example through development of separate modules by local developers. After proved competence in module development, local developers can participate in the core development. This follows the meritocratic approach of major open source projects.
- Later, during the operation phase with northern partners providing expertise and possibly still leading the core development, southern partners lead additional module development and take care of support for the installation activities.
- In the long term, lead development is not bound to the north. Coordination shall unfold through open source dynamics, built on experience gained by project contributors. Participants from the south use their knowledge to support new organizations in the installation and organizational implementation of the information system.

For the realization of a software project typically not only source code in the sense of programming output is required. Internationalization and reporting can also be important aspects. Translation and report template creation are possible entry points for initially inexperienced software developers and non-programmers. The steps of the ladder of increasing knowledge and local ownership can look like:

1. Understanding the working of a running system sufficiently to communicate user feedback to the requirements team
2. Ability to install, configure and maintain the system, to give basic technical support to users
3. Ability to build the system from a central repository and apply system upgrades
4. Ability to do construction tasks, for example (a) extensions for the local organization, (b) graphical report template construction, or (c) taking part in a team to construct certain functionality in a module
5. Leading the construction of a module given its specification, i. e. being responsible for a the design of a module
6. Defining requirements and distributing module design and construction to others
7. Gained competency and trust to improve the kernel or platform

The path of increasing knowledge goes along the reverse direction of the system development life cycle (see figure 5.3); The first learning experience can be to understand the operation of a production system. Further steps are learning about system installation, increasingly complex construction of technical artifacts, designing technical artifacts, and finally being able to formulate requirements. Thereby, the knowledge increase of less experienced developers is facilitated by the experience and the feedback by fellow, more experienced developers. The knowledge increase from operation skills towards definition skills also corresponds to the scale of increasing technological capabilities by Baark and Heeks (1999), as shown in table 1.1, p. 17.

Modular architecture

As outlined in section 5.3.2, modularity is an essential building block for open source software development. It facilitates decentralized development activities. The largely independent modules cover distinct chunks of functionality, and are integrated through a kernel module that provides basic functionality for the application domain.

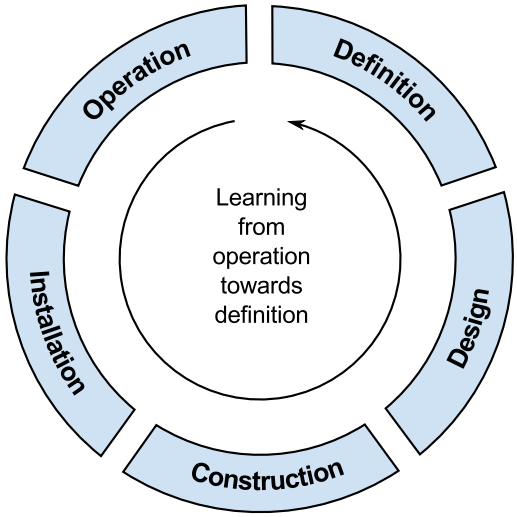

Figure 5.3: Learning IS development from operation to definition

OSGi (http://www.osgi.org) is an example of a framework that provides sophisticated modularity, which is in the process of becoming available also for web applications. The extent of the core module is typically limited to standardized functionality that is applicable to all installations. An example of a module with separate functionality is given by organization specific extensions, such as screens that support more efficient work activities in the specific circumstances of the particular organization. Such a module is ideally designed by the organization's local software developers.

Building on open source components

Another architectural approach, which is logical for open source software, is to base the system on third party open source components and libraries. A point to consider in this respect is the type of license of the used components, since not all licenses are compatible with each other. The licenses of the components may inhibit certain license choices for the overall system (see section 5.2.4).

Development focus at installation

The installation of the information system at an organization is meant here not only as the technical installation of the system on the server and client computers of a particular organization, but also to train the users and to streamline the organization's

work processes. The goal is to reach the point at which users consider the system an added value. Depending on the users' ICT skills, domain skills and other factors like the necessity to restructure organizational processes, the installation can take considerable time.

One of the factors influencing the installation process is the maturity of the system itself. Once software is delivered, it usually contains errors, which shall be fixed upon discovery (van Vliet, 2008). Similarly, weak usability may require improvements to the system before it becomes acceptable in the work environment. Therefore, some extra development efforts shall be foreseen for the installation phase. For improvements resulting from installation experiences, like in all further development activities, careful consideration is needed whether to carry out improvements in the global, standardized version, or in the form of local adaptations and extensions.

'Concentrated installation package'

To further underscore the significance of the installation phase for local learning and acceptance of the system in user organizations, the following is a recommended package of products and activities to be provided, called the *concentrated installation package*:

- Information on hardware and software requirements to run the system
- Full user and technical documentation in electronic form, but also copies of each in printed form, given a possible shortage of printing material
- Detailed scenario for the technical installation (description of steps, people to involve, time planning)
- Import of existing data
- Intensive training for technical support staff of the system (server administrator, database maintainer)
- Intensive user training
- Detailed description of the *best practice* work flow, taking into account the specific context (previous way of working) of the organization
- Training of the management concerning the effects, consequences and opportunities as a result of the installation of the system

The installation package should be worked out and executed for a particular user organization by the support organizations, in consultation with local representatives and the central leading authority.

Local support structure

The local support unit has the objectives to offer training on the use and technical maintenance of the system, to coordinate evolving requirements with software developers, and to be a point of reference for new organizations that are interested in the installation of the system. A local support unit may be part of a multilevel support structure. Ideally, most emerging problems are solved inside the organization in which they occur. Problems that cannot be solved internally will need to be answered by the local support unit. If this still cannot resolve the issue, such cases will be answered by the maintainers of the standardized software.

5.4 Chapter conclusions

This chapter has worked out a possible framework for collaborative information system development within globally distributed networks of organizations. The framework comprises the roles of local user organizations, support organizations, foreign expert organizations and donors. Furthermore, the framework involves a learning process, which starts from simpler parts like understanding the functionality of an operating system, to advanced skills like the specification of system functionality. Also a set of tools and methods has been proposed to guide the collaboration, which are related to the system development life cycle phases as identified in chapter 4.

Chapter 6

Managing change: How can local innovation be nurtured?

The previous chapter has pointed out the critical importance of balancing standardization and localization, of the installation phase as a decisive time to establish an information system's added value for the organization, and of the provision of user support in order to maintain the information system's relevance and usefulness. This chapter looks at the involved actors in IS projects, proposes a stable project structure and investigates various aspects of an information system change agent.

As will be elaborated in this chapter, a great facilitator towards successful IS development is the so-called change agent, who in general terms tries to bring together different actors from within and from outside an organization, thereby facilitating project progress. Possible scenarios in information system projects include a change agent who is present in a user organization, or a consultant hired from a support organization. The IS change agent can be described as an information system expert who links users and developers. He thereby facilitates IS innovation by assisting in its organizational implementation and by improving the IS artifacts, ranging from small local adaptations up to large scale improvements, which are of potential interest to other users of the system.

6.1 Stable project structure

In addition to a broad outline of actors as presented in figure 1.1 (p. 23), stakeholders can be further organized in the following groups, which relate to different phases

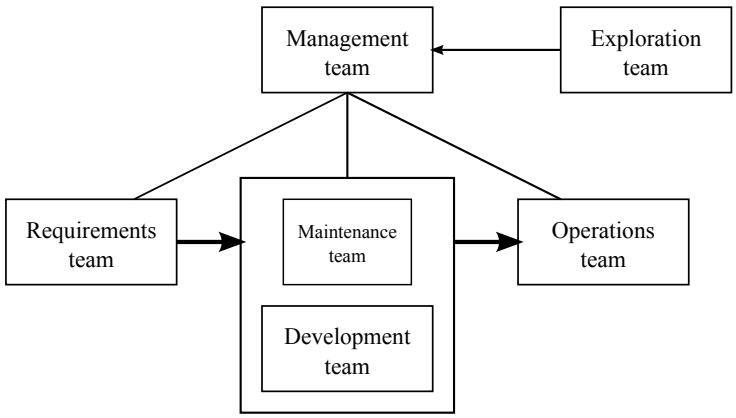

Figure 6.1: Stable project structure.

of the system development life cycle (see figure 6.1): the management team, the requirements team, the operations team, the development team, the exploration team, and the maintenance team.

6.1.1 Management team

The management team is responsible for overall management and financing issues. This team also handles the reporting to the funding organizations. In development cooperation projects this team includes both the northern and southern partner. Part of this team is the local steering unit. This is a subgroup to manage the local processes and another subgroup to manage the donor activities. The important role of the management team shall be emphasized. It is responsible for coordination between the other groups and to react to the often dynamic realities.

6.1.2 Requirements team

This team is responsible for the formulation of the requirements – and hence the functionality – of the system to be built. It formulates a requirements report and agrees on the system model as negotiated with the development team on the basis of this report. This group combines local and international expertise. Ideally, the requirements team has an international character, in order to achieve a good balance between standardization and localization. People from the requirements team may be characterized as domain experts.

6.1.3 Operations team

This team is responsible for the operations of the (new) system. It is aware of the local situation and can transform the local requirements into constraints and parameter settings of the information system. This group also provides the local infrastructure required for the IS, for example setting up a help-desk. This group preferably consists of local participants. This requires that universities adapt their programs to build these skills. Until that time, international partners can assist the operations team.

6.1.4 Development team

The development team is responsible for modeling the requirements provided by the requirements team, and to validate this model with the requirements team. Then, the development team will set up a system design in accordance with the operations team. This team may consist of the donor development partner, the local development team and maybe some subcontractor. The development team is also responsible for the installation of the system and to carry over to the operations team.

The development team has a local component that handles the reception of the new technology as set up by the donor partner. This component may also have been organized as a local community in the open source sense. Capacity building is an important issue during the project, to ensure skilled people are available for the development team. Only that way the local contribution can cope with an exposure to the open source philosophy. Until local skills are sufficiently present, the development team may *insource* from the international partner. Outsourcing experiences may be helpful to improve the success of insourcing.

6.1.5 Exploration team

The exploration team combines and shares best practices and finds opportunities to explore the system optimally. Typically, this team will consist of user groups. The exploration team has an important role to play in dealing with upcoming requirements on an ongoing basis. These requirements need to be prioritized and forwarded to the other teams for system improvement.

6.1.6 Maintenance team

Furthermore, there will also be a special team for maintenance. In many projects the maintenance team is by far the largest team. This team takes over development tasks that were previously the domain of the development team. Typical tasks of the maintenance team include responding to problems with the current system, preparing the introduction of new hardware, and making extensions for new options in the system. Additional care needs to be taken when updates of the software are made. The maintenance team does not test on real data on the production system. After a proper testing procedure, the software is handed over to the operations team.

The idea behind open source is to share maintenance and development. Moreover, the open discussion is expected to lead to a more stable architecture of the system.

6.2 Innovation and communication

As indicated in the previous section, a variety of groups need to work together to drive information system innovation. This section takes a look at innovation concepts and how change agents can facilitate the innovation process.

6.2.1 Feedback loops

Feedback has been identified as having a strong influence on the overall software development process. Not only the technical forward path of software development is decisive for the progress of the software development process, but also the feedback based on technical, business, user and other activities, making the overall software process a multi-loop, multi-level feedback system. Such feedback is important because it poses constraints on the possible progress of a software project. However, software development process models typically focus on the technical forward path and overlook the many feedback paths and their constraints on software based projects (Lehman, 1996).

The introduction of information systems into organizations leads to ongoing technical as well as organizational change and adaptation. This is illustrated in figure 6.2. Micro-level activities interact with higher-level phenomena. These macro-phenomena are called *accumulations*. Four different accumulations have been identified by Luna-Reyes et al. (2005): (a) system requirements, (b) system functionality, (c) organizational design,

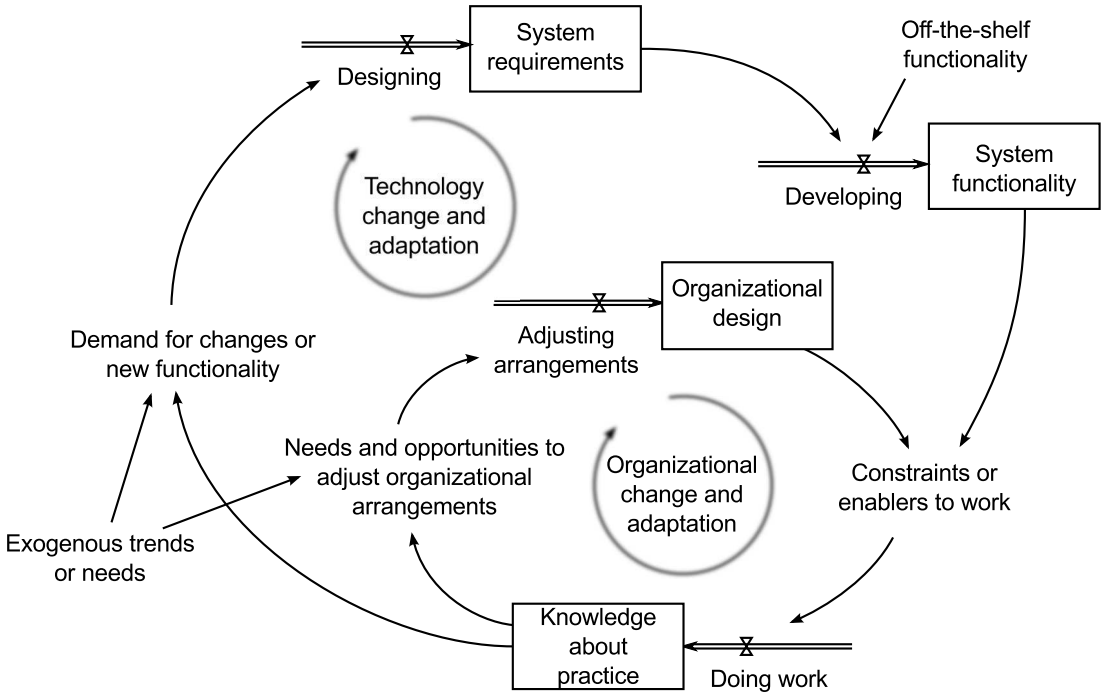

Figure 6.2: Recursive interactions among practice, organization, requirements and functionality (Luna-Reyes et al., 2005).

and (d) knowledge about practice. Two of the accumulations, system requirements and system functionality, are of a technical nature, while the other two, organizational design and knowledge about practice, are social.

Daily micro-level activities shape accumulations, and only changes in micro-level activities can alter the accumulations. The micro-level activities are represented by pipe-like arrows. For example, the way in which work is done, shapes the collective knowledge about practice. In turn, accumulations shape activities, either via organizational or technical change and adaptation processes. The four macro-level phenomena, or accumulations, are interrelated. The four accumulations change gradually over time, not in isolation, but by influencing each other. For example, new knowledge about practice may trigger demands for changed or new functionality. This is the beginning of a technical change or adaptation process, which leads to designing changed or new system requirements and the developing of the respective system functionality. With a new set of constraints or enablers for work, yet new knowledge about practice tends to emerge. Organizational change and adaptation processes work similarly. Over time, the four accumulations evolve.

This conceptualization represents information system development as an emergent process, where social and technical structures interact in a recursive way along a chaotic path, with feedback as a central component. Accordingly, project leaders may only partly be able to act as rational planners, but more like "surfer[s] on waves of change that focus on solving problems and meeting challenges as they emerge" (Luna-Reyes et al., 2005, p. 103). Learning and adaptation are basic elements in such an emergent process, in order to identify and manage, or even tinker with unforeseen situations. Rich feedback flows are a requirement for successful information system projects. If feedback does not find its way into the global software development process, there are serious implications for sustaining the system's utility, and hence relevance. Therefore, it is appropriate to put a focus on enabling feedback loops from user organizations to technical system developers. Effective user-developer communication cannot occur when users withhold feedback – especially negative feedback – about a system (Gallivan and Keil, 2003).

6.2.2 User innovation

Users play a decisive part in innovations. In many fields user innovation exists and is concentrated in the *lead-user* segment of the user community. This is the case for industrial products and processes as well as for consumer products. Lead users improve products or processes, or even invent new ones in order to meet personal needs (von Hippel, 2001). Also in the history of open source software development, influential projects often started out as solutions to personal needs of the author's everyday problems, and then spread because the solution is being found appropriate for a large number of users (Raymond, 2005).

In a wider sense, users can be conceptualized as either individuals or organizations who use a particular product, for example an information system. Users benefit directly from their innovation by using it, whereas manufacturers, such as product developers and service providers, must sell innovative products or services in order to profit from their innovation. Innovative users sometimes build or improve products on their own, without a manufacturer involved, for example in case when no product or service is available from any manufacturer. An illustration of the widespread existence of user innovation is given by a three year study in Great Britain, which shows that the amount of money that individual consumers spent on making and improving products was more than the amount spent on product research and development by all British firms combined (von Hippel, de Jong, and

Flowers, 2011).

A user network is formed by a set of user nodes that are interconnected by information transfer links through means such as face-to-face meetings and mailing lists. von Hippel (2007) contends that it is possible to build fully functional *horizontal user networks*, i. e. networks consisting exclusively of innovation users, if the users are self-manufacturers. Users in such a network invent innovative products for their own use. Then, these users share their innovation with other users, some of which on their part improve the innovation and reveal their improvements. An example of such user innovation networks are certain open source projects, particularly community initiated open source projects (see section 5.2.5, p. 140).

The point has now been made that users play an important part in innovation of products and services and that innovation is not the exclusive domain of manufacturers. However, innovation that is driven exclusively by users is an extreme case. Collaboration between manufacturers and users is a more common scenario in the overall innovation process. Non-users are also able to play an important role in innovation networks. For example, providers of complementary goods and services can be motivated to contribute to open source software innovation networks. In the case of developing country contexts, support services can be essential for user organizations to sustain their information system implementations. This is further elaborated in the subsequent chapter 7.

The distinguishing criteria between user-only innovation networks and mixed networks involving manufacturers is the question if users are able to self-produce and if they are competitive with commercial production (von Hippel, 2007). Not all user organizations in the developing country context will have the means to act as software producers for the information systems they use. However, such less technically oriented users still play an important part in the evolution of real world software systems, since a thorough understanding of the use context and recursive feedback interactions are essential. However, the transfer of this knowledge itself is challenging, as shown in the following when information stickiness is discussed.

6.2.3 Stickiness of information

Innovation in the sense of new product development or improvements to existing ones requires an accurate knowledge of the user's needs and the context of use. Otherwise

products may suffer from low relevance. The knowledge that developers need is generated at user sites, which makes it necessary for developers to get hold of this knowledge. This is not an easy task, because user information has a tendency to be *sticky*, meaning that there is a cost involved to transfer information from user sites to outside developers. For example, information about the conditions that make software crash are available for free at the location where the problem occurs, but can be difficult to reproduce at other sites. The knowledge transfer is additionally aggravated because users' needs and habits change constantly. For example, users change their knowledge about practice when experimenting with prototypes.

As shown in figure 6.2 (p. 163), the successful transfer of user knowledge about practice is important, since an ineffective transfer leads to the development of products that do not accurately meet user needs. In this case, users end up paying an *agency cost*; the manufacturer represents the agent of the users with respect to product development. The cost rises if information does not arrive efficiently at the developers. Hence, there is an incentive for users to facilitate the transfer of relevant information for innovation (von Hippel, 2001).

6.2.4 User-developer communication

Users need to take an active role in nurturing their innovation, particularly in the case when the market does not provide appropriate solutions to choose from. It is essential and beneficial to share innovation related information within the innovation network in order to match needs and opportunities.

User-developer communication plays a critical role in user participation in the software development process. However, this communication process is easily undervalued or taken for granted (Hartwick and Barki, 2001). It is often assumed that the transmission of feedback from users to developers *does* occur, and that this communication happens without problems, but these assumptions are questionable. It has been shown that important information is not always conveyed from users to developers (Gallivan and Keil, 2003).

To take advantage of expertise in distributed software development teams, rich interpersonal communication is a key enabler. It is not sufficient to know where certain expertise is located in the network, but the participants need to establish ways to apply the expertise to given problems in a timely manner. This is best achieved through an "emer-

gent process of informal interactions and joint problem solving" (Faraj and Sproull, 2000, p. 1557). Thereby, not only developers, but also users need to engage in the interactions, since users are the providers of valuable information about requirements and context.

Informal communication has been identified as a complementary way to coordinate software development projects. More precisely, informal communication complements the following approaches to improve manageability: Technical tools, modularization, and formal procedures. Technical tools improve productivity of individual developers, e. g. syntax highlighting editors. Modularization relates to both object-oriented programming and separation of requirements. Formal procedures include version control systems, delivery schedules and requirements documents.

Informal communication is frequently used in situations of uncertainty. Software projects typically incorporate a high degree of uncertainty. Managerial and technical problems continually arise, and not all of them can be resolved by consulting formal documents. Projects with stronger links between participants have been found to be better informed and coordinated. However, the possibilities of informal communication are limited by the relatively high transaction costs. Therefore formal, written communication is also necessary (Kraut and Streeter, 1995).

Users communicate with developers through various communication channels, such as user requirements meetings, bug reporting systems or user surveys. Such communication channels can be characterized by the following parameters (Gallivan and Keil, 2003):

Breadth: The number of users that have access to the communication channel. For example, requirements meetings with a group of focal users have a low breadth, whereas user surveys may include the entire user base and therefore have a high breadth.

Depth: This concerns the richness of the communication medium. Online bug reporting systems typically provide limited user-developer communication opportunities (low depth), whereas requirements meetings allow for rich exchange of information (high depth).

Fidelity: Some communication channels may suffer in their accuracy due to filters between the source of the message and the recipient, for example if user surveys are conducted by third parties and then forwarded to developers. This inevitably results in a loss of accuracy.

One-way/Two-way communication: Some channels, such as requirements meetings, are interactive, whereas other channels like user surveys are primarily a flow of infor-

mation in one direction.

User-developer communication can be conceptualized as a process model with four stages, two of which are focused on user cognition and communication behavior and the other two focused on cognition and behavior of software developers (Gallivan and Keil, 2003):

1. Users become conscious of messages to communication to developers, for example concerning requirements, suggestions and complaints
2. Users transmit messages to developers through one or more communication channels
3. Developers receive and interpret messages from users
4. Developers set priorities and take action, based on their interpretation of messages

As a process model, the four steps represent necessary, but not sufficient conditions for successful user-developer communication (Markus and Robey, 1988). Even if all steps are followed, success is not guaranteed. Each of the four stages has associated risks that threaten the effective communication between users and developers. In stage 1, users may not be aware of the real needs, opportunities or problems concerning information systems. In stage 2, they may not be aware of proper communication channels, or be unwilling to reveal important information, e. g. because of certain issues that are politically sensitive. When stage 3 is reached, messages received by developers may have been distorted on their way, e. g. through interpretation by developers or intermediaries. Finally, in stage 4, the prioritization of received messages may be misled, for example by focusing on technical issues to the detriment of more fundamental, underlying issues that may in the extreme case even question the advantage of a certain information system in its current configuration at all. Any of these potential types of communication lapses may jeopardize successful user-developer communication (Gallivan and Keil, 2003).

To acquire relevant information from users, sensitivity concerning issues of organizational culture and dynamics is essential. Therefore, particular significance has to be put on including persons in project teams that understand the implications of the change that a new system brings and who are able to create an environment in which users feel free to share their concerns, no matter how critical or sensitive those judgements may be (Gallivan and Keil, 2003).

Different names have been given to such a role. The names refer to a slightly different focus. Examples are the *social system analyst* and the *active question elicitors*. Social system analysts attempt to understand likely usage patterns and participate in system

design to fit the system to organizational needs (Gallivan and Keil, 2003). Active question elicitors are third party user advocates, who encourage users during the design process to ask more questions to strengthen their position and to weaken the power imbalance and semantic gaps between developers and users (Brabander and Thiers, 1984). Such user advocates may be able to raise the level of user influence throughout the life cycle. Creating the conditions for effective user influence appears to be a critical step to realizing the benefits of user participation, even though this might lead to conflict (Gallivan and Keil, 2003).

6.3 Change agents

A concept that is related to social system analysts and active question elicitors is that of a *change agent*. Part of the change agent's role to bring about IT-related change is to enable effective user-developer communication. The change agent concept is based on the recognition that organizational change is difficult, even if best practices are followed. To make matters worse, in many IT related projects best practices for change management are not followed, which easily leads to project failure. Often, it is unclear who is responsible to manage the change; a common view is that users are expected to benefit from the change, developers are producing IT artifacts that imply change, and managers initiate IT projects and set objectives for changed user behavior and organizational results. But in this conceptualization, none of the roles is in the obvious position to take charge of managing the process and deal with the implications of change. Managers tend to limit themselves to formulate targets, but not to accompany the process of change. Change is not entirely packaged in the IT artifacts either, but related to other dimensions also. For example, users may have other intentions than managers and developers regarding the use or non-use of information systems. Therefore overcoming hidden agendas can be a challenge. Hence, change is not automatic or straightforward. Another factor contributing to the challenge of effectuating change is the fact that current organizational practices are in place because they have been found useful in the past, thus changing them through the introduction of IT likely creates resistance. IT itself is unable to ensure that users will use it as intended. Only mindful use eventually achieves appropriate and effective use.

Change agents try to counteract the danger that everyone assumes that "change management is the job of someone – or something – else, [that] there is often no one left to perform change management tasks" (Markus and Benjamin, 1997, p. 66). The change

agent's role is to bring together the necessary conditions for IT-enabled change. This includes the difficult task of changing people's minds. The relevance of a change agent is underscored by the fact that for most people it is easier to learn about new work practices by talking about them with others who embody the new ideas than by reading about them. Therefore, personal contact is instrumental for change: "change is a contact sport" (Markus and Benjamin, 1997, p. 59).

Two different flavors of change agents can be distinguished: facilitators and advocates. *Change facilitators* try to enable informed decisions based on valid information. They enable rather than create change. The guiding principle is that people shall be empowered about IT, not by IT. The facilitator role is based on the belief that people, not technologies, create change. Because people have to create the change, they need to be empowered to do so. The facilitator aims to bring together good technologies, supportive organizational conditions and knowledgeable, mindful users. Facilitators are neutral and do not advocate a particular position or solution but mediate among those who do. Particularly in large-scale projects this role can be beneficial, by mediating among others without taking sides and advocating particular solutions. The facilitator role can be taken over by managers or IT specialists within the organization, or even outsiders such as external consultants.

Change advocates are deliberately political actors who try to change people's minds by any available means, including persuasion and exercise of power. They have their own vision and want people to follow them. Change advocates focus less on empowerment and more on inspiring people to overcome challenges. The guiding principle is 'whatever works'. The advocate quickly discards tactics that don't work. This role can be effective when difficult decisions have to be made, such as in large-scale IT infrastructure projects. Even though it is often assumed that managers would be ideal for the change advocate role, in practice they are reluctant to advocate change in IT projects. Effective change advocates can be found in every organizational role, including technical specialists (Markus and Benjamin, 1997).

Figure 6.3 depicts possible change agent scenarios in IS development projects like the OPUS project. Org. C represents the case of a strong user organization, which has their own change agent. However, not all organizations may be able to identify an internal change agent, they may for example lack the operational skills, or there may be tensions that are hard to resolve for insiders within the organization. In this case they may hire a change agent from a support organization (Org. B). This can be a service provided by

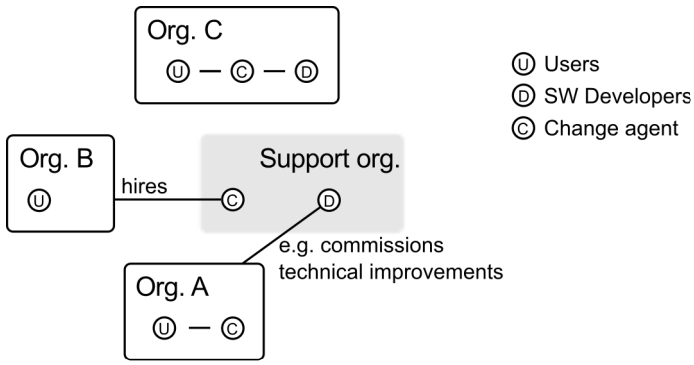

Figure 6.3: Change agent scenarios

the support organization, similar to offering software development for feature development or adaptations. Other organizations may have a change agent, but want to outsource software development activities to a support organization (Org. A). In this case, the change agent is an important link between the two organizations. In any case, the change agent's responsibility is to make change happen in a way that the user organization takes optimal advantage of the information system.

6.4 Chapter conclusions

- A variety of actors within and outside of an organization are typically involved in information system innovation projects. Special importance is related to the coordination of the various actors.
- Four interlinked phenomena are the cornerstones of a feedback-based, emergent process of information system development: System requirements, system functionality, knowledge about practice and organizational design. The phenomena are interlinked by technological and organizational processes of change and adaptation.
- The knowledge concerning user needs and context has to be communicated to the developers to maintain relevance. This doesn't happen automatically because information has a tendency to remain *sticky*.
- The communication process follows a four stage model of users identifying relevant messages to transfer, transferring the messages to developers, interpreting messages by developers and prioritizing actions.
- Both formal and informal communication links are instrumental to project coordina-

tion.

- User-developer communication in particular, and IT-enabled change more generally, need to be managed actively, otherwise nobody may feel responsible.
- Change agents aim to facilitate or advocate innovation by combining the right people, supportive organizational structures and technologies. The change agent position isn't necessarily best played by managers, but may be taken over by any organizational role.
- Change agents may reside inside the user organization, or they may be hired as third party consultants.

Chapter 7

Support: How can IS be supported in the long term?

This chapter intends to provide insight into possible approaches towards information system related support. Thereby, a community system viewpoint is adopted, which not only includes technology, but also covers people, knowledge, processes and support (Bieber et al., 2007). It is based on the notion that the *owner* of an information system is the community rather than a particular organization as in traditional Management Information Systems (MIS) research (Gurstein, 2007). These concepts are the basis to derive a framework for a supportive organization and a set of empowerment methods for the community participants.

7.1 Communities

This section explores the concept of *communities*. This includes an examination of community development and community informatics, particularly in the context of developing countries, and methods to empower communities. Furthermore, the basic processes are outlined that need to be carried out by a community in order to develop and implement information systems.

7.1.1 What is a community?

The term *community* has no single definition in the social sciences. Therefore it is necessary to define the term in every publication that uses it. One attempt that includes both physical and virtual communities is to characterize community as (1) a group of people, (2) who share ongoing social interaction, (3) with some common ties between themselves and other members of the group and (4) who share an area (common space) for at least some of the time (Hamman, 1997).

Another useful definition of the concept of community is that community is self-defining: it can be the *people with the problem* (Stillman, 2006, citing Stoecker), indicating a common objective shared among the community's participants. A similar definition states that "community is constituted by individual identification of and involvement in a network of particular associations" (Virnoche and Marx, 1997, p. 86). For example, individuals may form a community by working together on a project, sharing knowledge, making decisions or socializing.

Communities exist in different contexts, for example in family or work group contexts, as well as in different intensity of involvement. In any case communities have the function of enhancing the well-being of its participants (Bieber et al., 2007).

7.1.2 Community Informatics

Community Informatics (CI) is the application of information and communications technology to enable and empower community processes. CI is a framework for systematically approaching information systems from a community perspective, where the community is the *owner* or operative agent. This is an alternative to the traditional view that information systems are owned and operated by organizations. Community informatics has a practical orientation, as it typically relates to specific outcomes or actions in the world of practice (Gurstein, 2007).

Community informatics can be seen in contrast to approaches in management information systems, which has established best practices that generally assume an abundance of resources and expertise to which certain communities often do not have access. According to McIver (2003) the 'grand challenge' in CI is to develop technological solutions for communities that are economically, socially and culturally appropriate, and that are operationally and economically sustainable. This is relevant for developing countries, where

resources and training may be especially scarce.

7.1.3 Community Informatics and ICT4D

Conventional approaches to ICT4D tend to be dominated by a western, donor community set of values and priorities. ICT4D policies often follow a top-down philosophy that starts by defining national policy plans, followed by creating enabling conditions in the market, and finally creating projects that follow policy guidelines. This macro-level oriented ICT4D strategy does not necessarily provide access to individuals and groups on the micro level, and may thus prevent development opportunities in developing countries that would have been possible by more inclusive bottom-up approaches. This technocratic ICT development discourse has been emphasized by organizations like the World Bank. It has received critics because the approach excludes alternative views of technology and development (M. Thompson, 2004).

Vaughan (2006) positions CI as an alternative to the common top-down approach by referring to best practices and lessons learned from a plethora of case studies in the ICT4D field that suggest methods of CI – even though some do not explicitly call them CI. Community informatics proposes embedding ICT in existing community structures, utilizing existing social capital in those structures. Rather than imposing externally designed ICT solutions, ICT is introduced with the objective to help the community identify and meet its needs and to target effective use. Comparing the CI approach to ICT with the design and deployment of information systems in industrial contexts, the difference is that CI uses bottom-up processes for system design, whereas in industrial settings system design is guided by corporate management (Gurstein, 2007).

7.1.4 Clarification of terms: community development and appropriation

Community development seeks to empower individuals and groups of people by providing these groups with the skills they need to effect change in their own communities. For CI to be able to empower communities, it needs to fit into an overall community development strategy (Stoecker, 2005). ICT based and non-ICT based community development activities often run parallel and depend on factors such as the skills and preferences of the involved community members (Gurstein, 2007). Community appropriation of ICT designates a situation where the community has become sufficiently comfortable with the

technology to work both face-to-face as well as in technology enabled modes and decides on its own in which cases ICT is appropriate or not (Gurstein, 2007).

7.2 Social capital

Social capital refers to connections within and between social networks. There are multiple definitions of social capital, but a common understanding is that social networks have value for individuals or groups. In accordance with Yang, Lee, and Kurnia (2007), for the purpose of this study, social capital is defined as "an individual's network of social relationships and the qualities of those relationships [that] enhance the ability of participants to associate with each other for mutual benefits" (p. 231).

For Lin (1999) "the premise behind the notion of social capital is rather simple and straightforward: investment in social relations with expected returns. [...] Individuals engage in interactions and networking in order to produce benefits" (p. 30). Lin lists four elements that may explain why social capital works in cases where economic and human capitals do not. The four elements that give a possible explanation for the functioning of social capital are *information*, *influence*, *social credentials* and *reinforcement*. First, information available from social ties in strategic positions can enable individuals to know about opportunities and choices otherwise not available. Second, social ties may exert influence on decision makers. Third, acknowledged social tie relationships contribute to the individual's social credentials, which reflect one's access to resources that are beyond one's personal capital and may be useful to others. Fourth, social relations are expected to reinforce identity and recognition by being assured of one's worthiness as an individual and as a member of a social group sharing similar interests.

Portes (2000) argues for the distinction of two dimensions of social capital: individual and collective. Although "individual and collective benefits from primordial ties are not incompatible" (p. 3), social capital as a property of communities is qualitatively distinct from its individual version. At the individual level causes and effects are separate. Material and informational benefits that are produced through an individual's social network are separate from the social structure that produced them. "Collective social capital lacks this distinct separation" (p. 4).

Table 7.1: Four categories of social capital (SC) and ICT studies

	SC as dependent variable	*SC as independent variable*
Individual SC	Connecting social capital	Influencing social capital
Collective SC	Changing social capital	Enabling social capital

7.2.1 Social capital and ICT

Yang, Lee, and Kurnia (2007) reviewed studies about the relationship between social capital and ICT and organized them along two dimensions. One dimension refers to Portes' distinction of individual and collective social capital. The other dimension is concerned whether social capital plays the role of a dependent or independent variable in relationship to ICT. As a dependent variable the impact of ICT on social capital is concerned. As an independent variable, the impact of social capital on the development and use of ICT is concerned. The intersection between the two dimensions results in a matrix of four categories (see table 7.1). Examples for the category *Changing social capital* are studies on the role of ICT such as TV and Internet on social capital building in communities. An example of the category *Enabling social capital* is to put a support center in place, and by doing so social capital is created that influences users to better take advantage of ICT. An example for the category *Influencing social capital* is to give education to users of ICT.

The aim of this chapter is to identify activities, tools and methods that build and maintain social capital, mainly on the collective and to a certain extent on the individual level. These tools are supposed to positively influence the users' ability to develop and implement information systems. Hence, this corresponds to social capital being an independent variable in the categories *Influencing* and *Enabling social capital*.

7.2.2 Social capital and organizations

Cernea (1993) makes a strong case for the relevance of social capital for sustainability of development initiatives in developing countries. He argues that the social components to sustainability are no less important to development than the economic and technical ones. "Sustainability must be *socially constructed* – that is, arrangements of a social and economic nature must be made purposively" (p. 11). Where formal organization is weak there exists a high vulnerability to exogenous forces and less power to mobilize

social potential. To counteract this vulnerability and to facilitate enduring development it is necessary to form socio-organizational structures. Creating and strengthening adequate organizational structures and involving the users and other beneficiaries of the technology build the social capital that sustains, uses and maintains the technology.

> "Creating organizations is equal to creating social capital. Appropriate organizations are needed to enhance individuals' social capacity for coordinated action and empower them as agents of development activities. [...] In sum, promoting group formation and creating organizations are not easy social endeavors but are key avenues for 'putting people first' and for designing strategies around social actors." (Cernea, 1993, p. 13)

7.2.3 Sustainable CI initiatives from a social capital perspective

Simpson (2005) integrates social capital with theories of diffusion of innovation and community development into a theoretical framework for community informatics initiatives. The framework describes the interplay between physical infrastructure, soft technologies, social infrastructure and social capital. In this model, social infrastructure reflects Cernea's emphasis on organizational structures and includes furthermore social networks, local champions and other community resources. Soft technology includes education, project management, awareness raising, leadership building, financial and business planning, and conflict resolution. Social capital is manifested in common values and norms, in proactivity and leadership, in participation in internal and external networks, and in a strong sense of community. In this framework CI initiatives build social capital, and in turn social capital facilitates CI activities. Simpson's framework for social capital building is an example for the category *Enabling social capital* according to the taxonomy of Yang, Lee, and Kurnia (2007).

Regarding Simpons' framework, diffusion theory highlights the need for a long term view for innovative ICT projects in order to achieve widespread adoption beyond those participants that can be classified as innovators and early adopters. It further recognizes the role of local opinion leaders and change agents who can influence other community members' attitudes towards using the innovation. It emphasizes the creation of opportunities for emergent leadership.

Community development and capacity building focus on soft technologies such as awareness raising, training and local leadership building. These activities need to be inclusive across communities to both encourage participation and to support skill and confidence

development.

CI initiatives, through their combination of ICT and community development oriented practices, have a potential to facilitate rich communication – horizontal and vertical, as well as internal and external communication. As Simpson (2005) points out, this strengthening of both strong and weak ties is a unique contribution of CI activities to social capital building in the context of ICT.

The sustainability of CI initiatives is not only dependent on the extent to which the project addresses community needs, but also on the processes used to achieve them. Top down approaches that supply limited term funding may have only insufficient long term effects unless social aspects of the local environment are taken into account. Projects need to be designed in a way to incorporate soft technologies that build local capacity and leadership, encourage community ownership and strengthen local social infrastructure and networks.

7.3 Empowering communities

Schuftan (1996) provides a rough taxonomy of community development approaches and how they empower communities. He argues that community development actions are context dependent; the same action may sometimes be empowering, other times not. Rather than a single event, empowerment is a continuous process that enables people to understand, upgrade and use their capacity to better control and gain power over their own lives. Schuftan's taxonomy comprises the approaches service delivery, capacity building, advocacy and social mobilization (see table 7.2).

Service delivery provides a usually structured set of services to defined beneficiaries and addresses actions directly related to the immediate causes of maldevelopment. Examples include health and educational services. On its own it tends not to be very sustainable. Capacity building raises people's knowledge, awareness and skills to use their own capacity and that from available support systems to solve local problems. It strengthens the Assessment-Analysis-Action process in the community and thereby leads to more sustainability. Advocacy is about setting in motion a dynamic process of developing a consensus and a mandate for action. It brings together like-minded allies with a common goal. Social mobilization is the community development approach that gets people actively involved in development assessment-analysis-action processes. It engages them in actions to fight for

Table 7.2: Community development approaches that empower communities (Schuftan, 1996)

Service delivery	Using local human resources whenever possible
	Involving community representatives in the choice of delivered services
	Training local staff with the aim of behavioral change
	Assuring a continuous flow of information between providers and end users of services enabling the latter to be equal partners in planning, delivery, management and evaluation of the services
Capacity building	Enabling individuals, communities and organizations to continuously upgrade their ability to know, analyze and understand their situation and their problems
	Increasing people's awareness of what is permissible and fair to do
	Capacitating people to use explicit assessment-analysis-action processes
	Emphasizing skills that lead to community ownership of the interventions undertaken
Advocacy	Convincing and persuading people
	Increasing people's demand for, access to, and utilization of services, and their access to the means of production
	Promoting a more local control of resources
	Improving the access of end-users and facilitators to reliable community development-related information
Social mobilization	Articulating people's felt needs into concrete demands
	Networking with others, striving for achieving a critical mass of concerned people locally and externally
	Operating in complete assessment-analysis-action cycles, thus collectively identifying problems, searching for solutions and implementing them
	Giving people power over decisions, thus increasing their self-esteem and self-confidence

their rights and to gain more control over needed resources. It aims at networking, placing concrete demands and mobilizing resources. These community development approaches are inclusive to community participants and lend themselves to be integrated in organizational structures. Hence, they can contribute to sustainability of development efforts (Cernea, 1993).

7.4 Effective use

In his analysis of the digital divide, Gurstein (2003) proposes *effective use* as the goal to be achieved rather than simply access to ICTs and to the information society. Access on its own ensures opportunities to *consume* ICT enabled systems such as information systems, but it is a passive mechanism. It needs to be extended with or embedded in a greater context. In addition to access it is significant to have the knowledge, skills, and supportive organizational structures to make effective use of that access in order to achieve community objectives. For development to occur, access is a precondition, but the focus has to be on the whole development process, including infrastructure, hardware, software, and social organizational elements. Local communities need to train the capabilities to be producers, not just consumers, so that end users can do locally significant things with technology tools to which they have access.

Effective use occurs in social settings such as work groups and larger communities and is hence context dependent. What is appropriate in one context may not be in a different context. An example of a community informatics approach that supports local effective use is participatory design, where application design is done with full participation of the end users and the local community. In this way, an application is directly linked to local needs and creates local ownership and local champions who can provide feedback on its development and evolution.

Effective use is thus a goal of support and empowerment efforts for communities, because it fosters active community participants who increase their knowledge and become productive in the continuous improvement of the ICT systems they use to meet their needs.

7.5 Supporting and enabling communities

The *effective use* concept is the basis for a framework called *Supporting and Enabling Communities framework (SEComm)* (Bieber et al., 2007). It refocuses information systems towards communities and collaborative decision and design processes. It emphasizes system design that supports community members to become active participants in order to realize both collective and individual goals. It provides a model for reflecting on community support at all levels of the so-called *enabling community*.

Community systems include (1) technology, (2) people, (3) knowledge, (4) processes and (5) support. The design of community systems should specify people's roles – both those participating in the system's design activities and those using its outcomes or services; the kind of data and knowledge that should be acquired, stored and shared; the steps of the processes for accomplishing the system's purpose; and the support the system requires. Ideally, all participants should have access to the community's data and knowledge in a manner they can understand and utilize.

The SEComm framework consists of two elements: First, the *Participant Support System (PaSS)* encompasses processes, people, knowledge and technology in order to provide desired services and products. These elements are further influenced by environmental factors such as policies, constraints and the shared goal or purpose within the community. The second element, the *Community Participant Levels (CPaL)*, reflects the multi-level characteristic of communities and distinguishes between individual, group, community and supportive organization level. The different levels influence each other. At each of these levels a PaSS can be applied, and a PaSS at one level can influence another as environmental factor.

7.6 Analysis of the OPUS community

For an analysis of the community involved in the OPUS student information system project, the framework by Bieber et al. (2007) is followed to outline technology, people, knowledge, processes and support. First a short overview is given with relevant facts for the analysis.

7.6.1 OPUS community members

The common interest of the OPUS community is to manage academic data at institutions of higher learning utilizing appropriate electronic means. This academic data includes studies, students, exams and marks, taking into account the specific Mozambican reality and requirements.

People actively involved in this data management process are the academic registrars at universities and faculties in Mozambique. There is usually a chief academic registrar who can be characterized as a functional administrator for the area of academic registry at the respective institution. Other interested stakeholders include the university managers who need a data basis for decision making at faculty and university level, and the Education Ministry with similar intentions at the national level.

Academic registrars in the OPUS community have different backgrounds and experiences regarding the use of computers and administrative work. This heterogeneity means that learning and work progress vary strongly between users when computer based tools are introduced. For those with more difficulties in making sense of the new system it can be a motivation to be aware that others use a certain computer system and indeed consider the system useful.

Another observation is that time plays an important role in establishing a system. When difficulties arise and assistance is not (anymore) available, users tend to give up on new systems and go back to previous solutions. The relevance of time also comes into play according to the diffusion of innovations theory and its concept of the adoption rate of different adopter types, which states that innovators are much faster to adopt innovations than laggards (Simpson, 2005).

On the support side the ICT staff is important for the technical maintenance of the system. Furthermore, a help desk is useful for users to ask for assistance in case of any difficulties, and a software development unit is inevitable in the long run to develop the information system and adapt to upcoming changes in requirements (Lehman and Ramil, 2001).

7.6.2 OPUS community levels

Following Bieber et al. (2007), the following community levels can be identified in the OPUS north-south project:

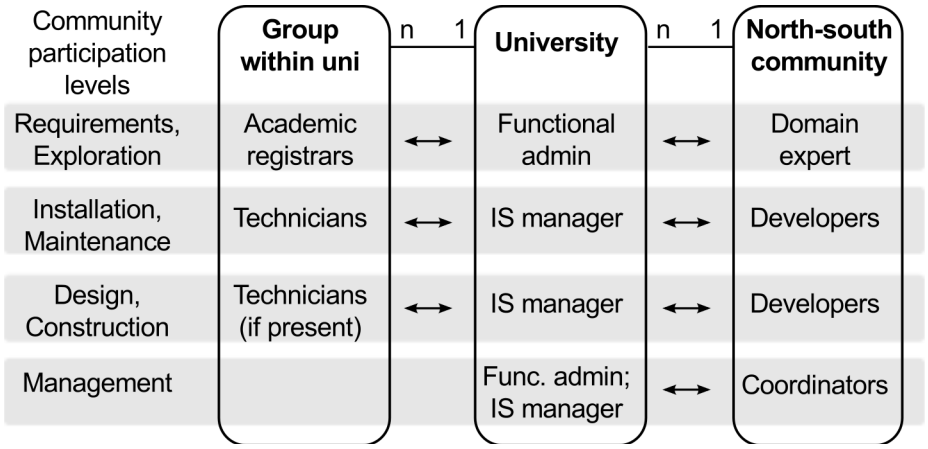

Figure 7.1: Community levels in the OPUS project

Individual level: Academic registrar, manager (i. e. users), technician, functional administrator, IS project manager.

Group level: Several individuals with similar interests form groups, e. g. the academic registrars within the university.

University level: All the individuals and groups within a university who work together to manage the university's academic data.

North-south level: Several universities together with a national government coordination unit and the partner in the north.

Figure 7.1 shows the project participants during the north-south project. The overall community consists of individuals forming functional groups such as the academic registrars within a certain university. This group is coordinated and supported by the responsible chief academic registrar who is the functional administrator for this subject area. The community furthermore includes participants specific to the north-south cooperation and outside the universities who either come from the national government coordinating body or from the partner in the north. During the north-south project these different participants together engage in the processes of the system development life cycle, such as requirements engineering and design as well as project management.

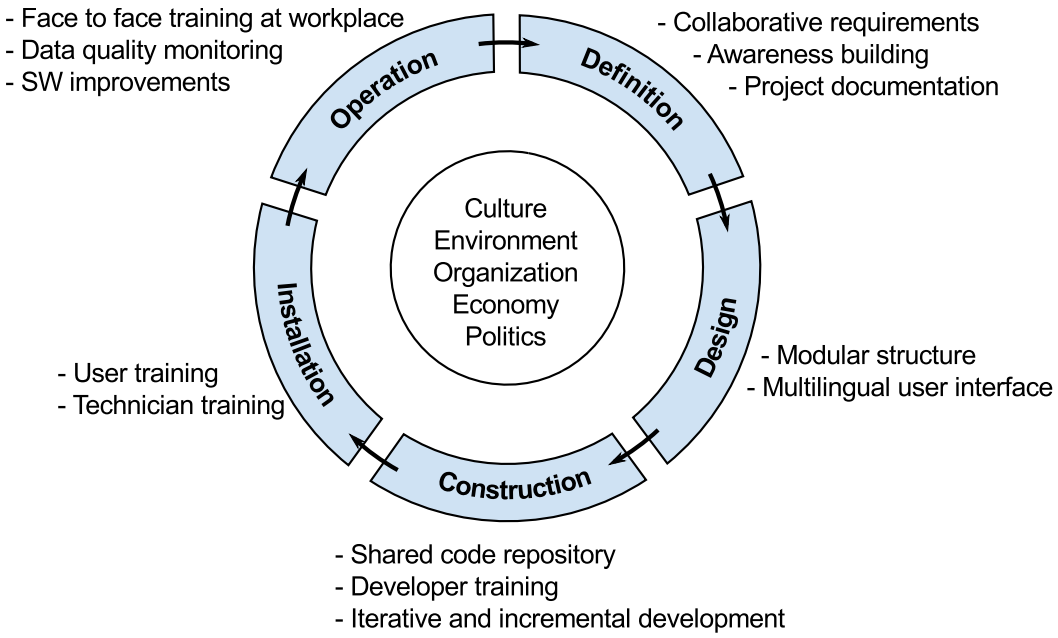

Figure 7.2: Empowerment methods applied during the OPUS project

7.6.3 Empowerment practices applied in the OPUS project

Figure 7.2 provides an overview of activities to empower the community as they were used in different stages. The Appropriate ICT framework life cycle model (see figure 4.1 on page 118) is used to organize the activities. This form of representation is not meant to suggest that a singe pass waterfall was followed (Larman and Basili, 2003). In fact, an iterative and incremental development approach was applied, and requirements were refined every time the participants from north and south were able to meet. Thus, activities like collaborative requirements gathering were an ongoing activity during large parts of the project. The activities are shortly described in relation to their empowerment aspects.

The activities were planned by the coordinators of the north-south project in collaboration with the participants of the universities. The definition phase began with awareness building involving university rectors and handout of initial printed project documentation. A website was created for further documentation and information publication. The requirements engineering process was an important method to actively involve university participants such as the academic registrars. It proved to be a challenge to get the complete picture of requirements, in some cases due to users' inexperience with computer

software.

In order to verify and adapt requirements throughout the design and construction process the iterative and incremental development approach was used where the users reviewed the progress at regular intervals when participants from north and south came together. During later stages local software development was done at one of the participating universities. This process was characterized by proximity between developers and users, which made it possible to quickly react to the discovery of bugs and to make appropriate user interface improvements. This local software development followed agile software development practices, involving users as part of the team, focusing on small improvement cycles with short times between decision making and seeing the decision's consequences (Cockburn and Highsmith, 2001).

The student information system needed a user interface in the local language. The system's modular structure, together with a shared code repository and mailing list, enabled the adaptation and translation of the system by local participants. This, in turn, provided a possibility for local learning and local ownership, based on a two-way communication between south and north. To foster local involvement in the software development, an extensive developer training was provided. The experiences were however mixed; the success of this method was limited due to unavailability of some of the local software developers once they were trained.

To address difficulties during the installation of the system at the universities, user and technician trainings were accompanied by visits of the project coordinators and developers to the participating universities. These visits also served to assess areas for further training in ICT, such as Linux and Samba.

Towards the end of the north-south project the universities needed to get more actively involved with the system. In the operation phase good experience was made with working face-to-face with users at their respective workplace. This time-intensive method did not only educate users how to use the system effectively and efficiently, but also identified difficulties with the system that were not obvious during earlier requirements gatherings and trainings, and cultivated a culture among users of actively pursuing support, which in some occasions resulted in improvements to the system. A related method was the monitoring of the quality of data that was entered by the users, which increased users' awareness of best usage practices.

7.7 Community support model

In this section a model is worked out on the basis of the analysis of the community system of the OPUS student information system development project. For this model several community participation levels are distinguished, and a supportive organization is outlined that is guided by the provision of empowerment activities.

As an information system development project moves from a sponsored north-south cooperation to a self-sustaining project, the activities and support given by the project partner in the north and the government unit in the south need to be organized in a different way, particularly where support is concerned. The goal in this section is to develop recommendations for a supportive organization that is able to empower the community participants, including institutions that already use the system as well as those interested to use it in the future.

7.7.1 Empowerment activities

For IS development, service delivery includes activities concerning the optimization of the functionality of the system such as further system development, adjustment and data import, and basic training of users and technicians.

Capacity building involves activities aimed at an adequate transfer of knowledge and skills to enable effective use by users and university management.

Advocacy includes activities to promote and create a positive attitude towards the system. An example is to raise awareness to potentially interested institutions who do not yet use the system.

Activities regarding social mobilization are aimed at creating interaction among system stakeholders. It is the basis for continuous system development, by involving end-users to collectively identify and address current problems and upcoming requirements. Change agents have the potential to play a strong role for social mobilization.

Table 7.3 provides an overview of possible empowerment activities by the supportive organization. A particular method, which has the potential to cover several community building areas, is open source. It creates capacity by offering community participants the possibility to learn and create local ownership. It advocates the system through clear license terms and free access to documentation and source code. It fosters social mobilization by allowing participants to get actively involved in assessment-analysis-action processes for

Table 7.3: Possible empowerment methods provided by the supportive organization

Service delivery	User help desk
	Assistance with system installation
	Documentation and manuals
	Translation of documentation and user interface
	Software development (e. g. additional functionality)
	External IS hosting
	Importing existing data
	User and technical training
Capacity building	Face to face sessions at workplace
	Establish effective use of the system in the institution
	Collaborative agile software development (e. g. open source)
Advocacy	Keep community participants informed about system updates
	Maintain project website with news and downloads
Social mobilization	Change agents
	User group meetings
	Steering group of community participants
	Collaborative requirements gathering for system improvements
	Mailing lists and on-line forums

system improvement.

A support organization may be an important source for more complex technical system improvements that exceed the capabilities of a single user organization. This can, for example, be the development of additional software modules, or the design of report templates and statistics.

The import of existing data can provide a boost to get going with a new IS and quickly make its advantages visible. It can substitute tedious manual data entry.

Face-to-face sessions with users are a form of training that focuses on resolving real-world problem situations. When users encounter difficulties in their daily tasks, it is decisive to have proper guidance to prevent frustration on the part of users. Establishing effective use of the IS within the institution emphasizes the use of the system in its intended way, the adaptation of organizational processes and the promotion of ownership and active engagement in system improvement processes.

Change agents can play an effective role in bringing together different actors primarily within the organization, but can also with outside partners. They are able to nurture conditions so that organizational change is possible. They can help to avoid that projects come to a standstill due to missing resources, misinformation or conflicts.

7.8 Chapter conclusions

This chapter has considered the community concept, community informatics and community development. Community can be broadly seen as consisting of people who want to solve a similar problem. Community informatics puts the local community members in focus. A central approach of community informatics follows the effective use concept, whereby not only access to ICT is to be achieved, but also the active engagement in using ICT for problem solving. Based on the analysis of the Mozambican OPUS community, a set of community development methods has been proposed, which are oriented towards local empowerment. Furthermore, the community development methods strengthen social capital by enhancing the individuals' social capacities. The improvement in social capital contributes to sustainability.

Chapter 8

Structurational analysis of cross cultural IS development: What constitutes success?

This chapter aims to produce a detailed understanding of the dynamics of the OPUS student information system development project, which has involved teams with African and European backgrounds. This is done by applying a framework for structurational analysis in cross-cultural software production and use. The analysis uncovers actual and potential conflict, and the cultural heterogeneity that exists between groups of participants. Furthermore, it reveals diverse measures of success by these groups, such as different prioritization of local involvement. Finally, it offers a dynamic conceptualization of how culture can be either reproduced or produced in new ways through human action, and thus brings to attention both opportunities for as well as barriers to change. This chapter concludes that implications for information systems in developing countries include a focus on managing change, incremental development and persistence over time, backed by committed leadership.

Information and Communication Technologies (ICTs) and more particularly Information Systems (IS) are considered by many authors as relevant to developing countries (Heeks, 2008). However, strategies for introducing IS that have worked in the developed world do not necessarily translate well in developing nation contexts (van Reijswoud, 2009). There are many examples of failure and partial failure, and the challenge is to understand

the difficulties and overcome them (Walsham and Sahay, 2006). IS-based innovations need to pay attention to the context within which they are embedded (Avgerou, 2001). Interpretive methods aim to produce "an understanding of the context of the information system, and the process whereby the information system influences and is influenced by its context" (Walsham, 1993, pp. 4-5). Prasad (2009) recommends a structuration perspective to understand the relationship between ICT and organizational performance in developing countries.

The analysis in this chapter is motivated by the assumption that in order to achieve success in cross-cultural IS projects, it is beneficial to understand inherent contradictions that may eventually lead to conflict, and to subsequently search for ways to manage or avoid them. Different measures of success by different actors are a potential area of conflict. In the complex context of cross-cultural work groups a wide variety of viewpoints is only natural. In many situations this can be enriching, but there is also a high probability of potential conflicts.

For analyzing processes of change during the application of ICTs, two dimensions can be combined (Avgerou, 2001):

- A horizontal analysis of the sequence of events over time
- A vertical analysis of higher and lower levels of context, e. g. from the international level down to the group or even individual level

These two dimensions will be combined in the chapter's analysis; the aspect of time is not only related to the sequence of events, but also to the actors' ability to reflect upon and change one's behavior over time. Vertically, three levels are being considered: the international level covering the project on a global level; the level of the universities, and the level of actors within a specific university.

The chapter is organized as follows: First, some observations are made concerning the research method for this chapter's analysis. Subsequently, the theory of structuration is explained. Then, the project is analyzed using the structurational analysis framework of Walsham (2002). Afterwards, the findings are synthesized in order to highlight different success measures and discuss implications for IS projects in developing countries. Finally, contributions and conclusions are elaborated.

8.1 Notes about the research method

During several years of action research a vast amount of observations has been made, and interpreting the data is inherently subjective. Therefore, this section pays attention to how this study was conducted, partly by referring to the principles of Klein and Myers (1999) for interpretive field research. Klein Myers mention that their principles shouldn't be used 'a la carte' and that each principle may vary between studies. Nevertheless, the principles can be used to ensure high-quality interpretation of qualitative data (R. Weber, 2009).

The principle of the hermeneutic circle is considered by Klein and Myers (1999) as the most fundamental principle upon which the others expand. According to this principle, understanding a complex whole is achieved by iterating between considerations concerning the meaning of the parts and the meaning of the whole that they shape. This iterative form of understanding applies to a wide variety of objects such as the interpretation of written text. In fact, it is suggested that all human understanding is achieved in this form.

Previous action research cycles influenced the project and produced certain understanding of specific aspects relevant to the project. These cycles followed the model of McKay and Marshall (2001), pursuing both research interests and problem solving interests, as outlined in chapter 3. An early assessment of appropriate technology for IS in developing countries had an impact on the project by emphasizing the local ownership of IS projects and the usability of resulting systems (see chapter 4). Furthermore, involving local programmers in a collaborative software development structure such as open source was seen as a way to build local capacity. Another conclusion concerned the importance of modular system architecture (see chapter 5). Moreover, the potential of local change agents on capacity building was recognized (see chapter 6). Further research focused on long-term IS support, which highlighted institutionalization of support structures as a way to nurture empowerment and social capital (see chapter 7).

In the following, some examples are given to illustrate how Klein and Myers' principles have been applied. Specific characteristics of the conduct of this study include the rather strong involvement of the researcher, i. e. myself, into practical project activities, and my European heritage. Both affect the interaction between researcher and subjects. Researchers as well as all other participants can be seen as interpreters, "as they alter their horizons by the appropriation of concepts used by IS researchers ... and other parties inter-

acting with them" (Klein and Myers, 1999, p. 74). This has been particularly true in the OPUS project, since I gave my input to, and thereby influenced, different people involved in the project on various topics such as software development, collaboration between distant participants in Africa and Europe, and on the implementation of the IS in Mozambican universities. To report one's own role as an involved researcher is particularly challenging (Walsham, 1995b). But at the same time this deep involvement prompted many rich observations during the project, as the research gained an insider view and participated in day-to-day activities (Prasad, 2009).

There have been a variety of different groups involved that are relevant for the analysis in this chapter. First of all, there were five Mozambican universities with different forms of institutionalization: public, private, and some with a religious affiliation. Within the universities, different groups could be identified such as managers, users and technicians. Additionally, there was a technical team from the Netherlands, and an international project coordination team with members from both countries. Furthermore, *aid workers*, i. e. consultants with a European background such as myself, assisted in introducing IS at UCM. Each of these groups had distinct interests and behaviors. This relates to the field study via Klein Myers' principle of multiple interpretations, which emphasizes the consideration of multiple viewpoints in contexts involving multiple agents, the underlying reasons for these different viewpoints and how they may lead to conflict. As an action researcher I had regular contact with all of these diverse project groups, which helped to understand and compile their distinct and often divergent viewpoints.

Data was primarily collected through participant observation and informal interviews. In contrast to Western culture, written documents play a less important role for project coordination and management in the local context. Mosse and Sahay (2003) noted in a case study in the health sector in Mozambique that co-location and face-to-face interaction are the predominant means of communication. Even though their study was located in a more rural setting, the same tendency was also observed in this project. Hence, despite the existence of some formal project documentation, data was gathered mostly through direct communication, as well as during project workshops. Furthermore, relatively new forms of communication such as email and Skype also played a documentary role.

8.2 Structuration theory

Structuration theory is a general theory of social organization, which has been promulgated by sociologist Anthony Giddens (1984). Is has been used extensively in the area of IS research, even though one of its characteristics is "the almost total neglect of the technological artifact and its abstract" (Jones and Karsten, 2008, p. 128). Nevertheless, authors such as Orlikowski and Robey (1991) have applied structuration theory to the analysis of IS. They argue in favor of the relevance of structuration theory for IS research by referring to its potential to integrate subjective and objective elements of social phenomena. They theorize information technology to be both "the product of human action as well as a medium for human action" (p. 144). Information technology both effects social structures and is shaped by users and developers. Walsham, who has richly applied structuration theory to IS, formulated a structurational analysis framework of cross-cultural software production and use (Walsham, 2002). This framework, which is applied in the following sections, considers four key areas to gain an understanding in such settings: (1) structure, (2) culture, (3) cross-cultural contradiction and conflict, and (4) reflexivity and change.

Structuration theory's central concept of *duality* states that human action (agency) and social structure are not separable, but are two aspects of the same, mutually constitutive, whole. Human action is guided by social structure, and action either reproduces existing structure or produces new social structure (Jones and Karsten, 2008). In structuration theory, *structure* is described as rules of behavior and the ability to deploy resources, and it exists in the human mind itself rather than as outside constraints (Walsham, 2002). For the analysis of structure, three dimensions can be distinguished: systems of meaning, forms of power relations and sets of norms. Structure and IS are interlinked: "IS are drawn upon to provide meaning, to exercise power, and to legitimize actions. Thus they are deeply involved in the duality of structure." (Walsham, 2002, p. 362)

The second key point, *culture*, is related to the fact that structure does not exist in the mind of a single person in isolation. People of the same cultural group share structural similarities, i. e. similar systems of meaning, forms of power relations and sets of norms. Despite considerable variation among people belonging to the same culture, there is enough *systemness* to be able to recognize shared symbols, norms and values.

Cross-cultural contradiction is based on another concept of structuration theory, structural contradiction. Contradiction is the potential basis for conflict due to cultural

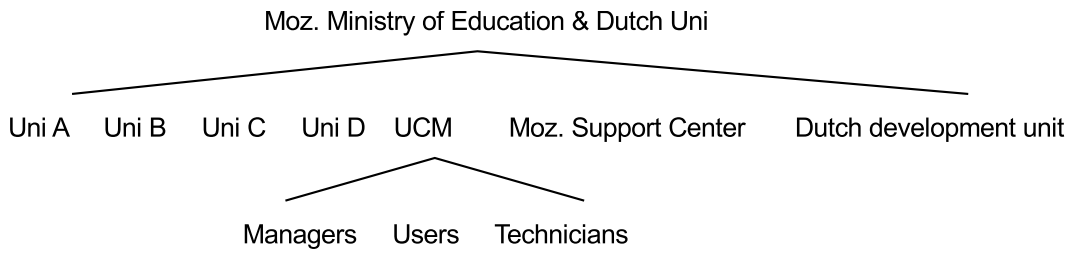

Figure 8.1: Project participants at three levels

differences, whereas conflict is an actual struggle between actors. Conflict may result if actors feel that contradictions affect them negatively and if they are able and motivated to act.

Finally, *reflexivity and change* refer to the observation that human beings do reflect on their own actions and those of others, and on their intended and unintended consequences. Such reflection permits human beings to either reproduce existing structures or produce new structures. Shifts in the minds of individuals and groups account for culture's dynamic nature.

8.3 Levels of analysis

The project participants at different levels are depicted in Figure 8.1. They are grouped into three levels. The international project coordination team consists of participants from local and external project partners. The middle level consists of the involved organizations. The lower level is concerned with groups within universities. This level is concerned specifically with the users, technicians and managers within UCM.

The analysis in the following sections is concerned with various project groups. While many differences can be expected between participants of African and European heritage, structuration theory makes it possible to identify groups with distinguishable interests and meaning systems, not only on the level of national culture, but also on a smaller scale than between nations (Walsham, 2002). First, the global level of the project will be considered regarding the cooperation between Mozambican and Dutch actors. Then the focus moves to the level of Mozambican universities, and finally to one particular university, UCM. An overview of the issues at different levels is shown in table 8.1.

Table 8.1: Overview of key issues of structurational analysis at three levels

	Overall project	*Universities*	*Within the UCM*
Structure and culture	Dutch team Logic of intervention projects Mozambican team Face-to-face communication Many actors Producer (N) – consumer (S) perceptions	Variety of participant backgrounds Excitement about participation Wary of external consultants New system calls for additional work Source of funding Org.-specific requirements	Autonomous faculties Registrars used to know students personally Learning of 'currently' needed features only External consultants
Cross-cultural contradiction and conflict	Hesitation to 'get hands dirty' Understanding requirements Transfer of source code Extent of documentation Ownership of source code	Activities parallel to project Conflicting requirements Desire for quick solutions SW developers trained but lost to project Short implementation phase	Centralization of student records Usability Incident-based data collection Roles of users and technicians
Reflexivity and change	Joint workshops opportunities for reflexivity Two-way learning between N and S Stronger focus on implementation	Inter-university discussions	Intensive face-to-face sessions at workplace Feedback cycles IS institutionalization

8.4 Macro level

8.4.1 Structure and culture at macro level

Dutch team

On a global project scale, the meaning that was associated with the project by the European team was to develop and deliver an agreed set of technical artifacts, capacity building activities and training of local participants. This understanding followed from the early phase of project definition and negotiation, which had resulted in a written and signed agreement between the two sides, and the formation of a project management team consisting of representatives of both sides, responsible for the execution of what had initially been agreed upon. The relatively small Dutch team comprised the project manager and a group of software development professionals. This team, in order to lead the development of the student IS, regularly visited the Mozambican participants and used their knowledge of structured software development processes to create the desired technical artifacts. Opportunities for participation were made available through workshops for collaborative requirements engineering and discussions about incremental versions of the emerging software system. Furthermore, they used these encounters for repeated training sessions to local users and developers.

Written form of communication

The approach to project execution by the European team was consistent with European business culture, i. e. the norm of fulfilling one's side of the contract on time, by providing the agreed deliverables. Furthermore, their work routine made regular use of written documents and emails to communicate project related information. In terms of power relations, the Dutch project manager had the ultimate responsibility and had to make final decisions, but also sought the input of the technical team members in their areas of expertise.

User interface focus on detailed information

In the design phase of the software system, a usability expert was consulted for the basic structure and layout of the user interface. This resulted in a consistent interface design. However, one problematic result was that the screens closely followed the database

structure. There were a large number of input fields, particularly the information concerning students and their study plans. In order to structure these fields, the user interface design provided groupings of fields into several tabs. There was no effort made to configure the visibility of fields, which resulted in users being overwhelmed and confused by displays of irrelevant fields. In defense of this design, it was argued by the Dutch developers that information that was not considered relevant at a certain time may become relevant later, and showing all possible options would offer users the power to work with that information at any time in the future.

Documentation of architecture

Documentation was considered a key element in the construction of the software system and was published on the project home page. In order to make the installation as easy as possible for Mozambican universities, step-by-step installation guides were written for different target operating systems. Use cases were documented, and a user manual and an overview of the database structure were provided. Database and source code documentation were limited to basic information, so that the continually evolving source code and database structure would not become out of sync. Furthermore, programmers who intended to contribute were expected to be able to interpret the source code.

'Installation equals implementation' approach

Implementing the IS at the universities was a short-term activity. Technicians had been trained several times during the project on how to install the system, detailed installation instructions had been provided and the users had been repeatedly trained on the system's features. This focus resulted in the perception that the system would be immediately ready to use after its installation. Furthermore, definition, design and construction took up most of the project time, so that little time was left within the sponsored project to carry out more effective integration of the student information system into organizational practice at the different universities.

Symbioses between interventions

Intervention projects are particularly appealing if their results can be reused in other interventions. It indicates efficient use of the funds that go into the project. The soft-

ware system that had been developed for Mozambican universities during this project was considered for a follow-up intervention project in Zambia. The system would be extended to include required features and implemented at two large Zambian universities. Ideally, the project would result in an open-source framework upon which further collaborative evolution of the software system with the parties from Mozambique, Zambia and the Netherlands would be possible.

Mozambican team

The Mozambican team was more heterogeneous than the Dutch team. The Ministry of Education coordinated participants of five different universities from all parts of the country and tried to establish a support unit that would assist Mozambican universities with the implementation of the IS in their institutions and with technical support. A functioning support unit was considered a crucial task with respect to the sustainability of the IS after the end of donor support. A prevalent idea of local participants was that the foreign experts would deliver the solutions to local problems within the framework of the intervention project. An illustrative example is that of computer hardware, which is a technology produced abroad and transferred to local organizations. Software was similarly seen as an artifact to be delivered ready to use at once. Because the development of the software system was an activity that spanned a large part of the project duration, the local participants found themselves in a *waiting seat* position – waiting for the final software product to be finished before getting involved.

Face-to-face communication

Face-to-face was the predominant local form of communication. This became visible in the regular joint workshops. Such encounters tended to be constructive with lively discussions taking place, facilitated by a general willingness to speak out in group settings. This led to a positive attitude toward the project and the system. Similar observations were made in personal talks with future users when assessing their opinion about the introduction of the new IS and bringing to their attention some possible consequences for their daily work, such as the extra workload during the transition phase. Such talks were usually positive, especially if local participants were assured of continued support and could sense that the implementation was a serious effort by university management. One academic registrar

stated that the introduction of the new IS "is indeed what we need and I will work very hard to make it work at my faculty, even working extra-hours if necessary." Another user who had initially almost no computer literacy skills, when given a careful explanation that all registrars would have to use this new system in the future, reacted by saying: "Sometimes it's necessary to move with the times."

There have also been more skeptical registrars, such as the one who was afraid to store sensitive data about students and marks on anything other than her own USB flash drive which she only attached to her computer when the network cable was unplugged. It was against her understanding of security to store data in a web application that is out of her control. Attempts to convince her by outlining other dangers that she was confronted with – such as the possible loss of data on her flash drive – failed.

Enforcing power was difficult for the Ministry of Education. This was because the Mozambican participants belonged to different organizations, and the Ministry had a coordination role, but no formal authority. Much depended on the motivation of individual participants, for example, to do their *homework* between workshops. The Ministry began to organize a countrywide support unit in order to gain more control. This unit is to be located in the computing center of Mozambique's longest standing university, Universidade Eduardo Mondlane (UEM). This initiative was still an ongoing effort at the time of writing.

Ownership of project output

The project agreement stated that all project output would become the property of the Mozambican side. This was seen as important to the project coordination team at the Ministry of Education. Rather than simply having access to the source code, the source code repository was expected to be moved to the support unit, with exclusive access to the source code.

Expect 'full' documentation

Mozambican software developers wanted detailed documentation of the database and source code. This was seen as a precondition to the understanding of the inner workings and their ability to be able to contribute to the system's evolution.

8.4.2 Cross-cultural contradiction and conflict at macro level

Little effective use

Despite the intentions by the European team to involve local participants, there was a barrier to this involvement. The Dutch project team's intention was to involve the Mozambicans with common workshops, and to charge them with assignments between the workshops. The Mozambican actors held the view that there was not much to be done until the system was ready to be used in production mode at Mozambican universities. This prevented local users from experimenting with the system, thereby developing an in-depth understanding of its logic and its fit with each university. Such ongoing experimentation would have been beneficial in providing feedback and influencing the development of the system, for example, by discovering missing features or weak usability. Another factor that jeopardized local skill building was the short implementation period, which left the users on their own to discover how to use the system features properly in their daily work.

The delay in getting local participants involved – *getting their hands dirty* – also limited the their ability to assess the workload required for successful system implementation at a university. This was evident in the formulation by over-ambitious plans, such as the Ministry's plan to introduce the system at 15 different Mozambican universities within 1 year.

Understanding requirements

Getting the correct understanding of requirements was challenging. The joint sessions between local and foreign participants were the basis for the construction of the technical artifacts. However, local participants had little experience in expressing requirements. Therefore, it was hard for them to imagine alternative ways of working compared to the current established routines. Sometimes, opposing requirements were stated, which could also be attributed to the yet little experience in providing IS requirements. Additionally, there was a natural language barrier between English and Portuguese. This made it harder to understand requirements and more likely to misinterpret statements. Furthermore, out of the set of desired functionality the most important had to be selected. Despite the prioritization, a lot of functionality was developed within the project, which prolonged the construction phase to the detriment of the organizational implementation phase.

Transfer of source code and documentation

While one group viewed the emerging software system as something that is always ready to be experimented with, the other side held the view that at one point in time, when work on the software system is finished, its source code and full documentation will be transferred to Mozambique like a physical artifact. There were also conflicts regarding appropriate amount of documentation.

Ownership of source code

The logic of the Dutch intervention project has been to maximize impact and assist as many potential beneficiaries as possible. This conflicted with the Mozambican position of owning project output, because the Dutch team intended to implement the software system in a different project in Zambia.

8.4.3 Reflexivity and change at macro level

Communication of IS innovation is a process in which participants share information in order to reach a mutual understanding. It is a "two-way process of convergence, rather than a one-way, linear act in which one individual seeks to transfer a message to another in order to achieve certain effects" (Rogers, 2003, p. 6). During the project, there have been many meetings among project participants in workshops, training sessions and coordination meetings. Most of these encounters took place in Mozambique, and a few, such as an intensive developer training session, were done in the Netherlands. These meetings provided opportunities for change – in the vocabulary of structuration theory these were opportunities for the production of new structures of meaning in the minds of participants.

Local actors started to engage with the idea of collaborative software development using a shared source code repository. Open Source was discussed as a means to resolve the ownership dispute. The database-oriented screen design was recognized as not being an ideal solution and possible improvements to the usability were considered. Moreover, for the follow-up project at Zambian universities, the Dutch project team already put a stronger focus on implementation and planned to install the system early, in order to be able to react to feedback from local users, technicians and managers.

8.5 University level

8.5.1 Structure and culture at the level of universities

The universities that participated in the project were heterogeneous organizations, and the actors involved had a wide variety of backgrounds and motives for participation. Some had already gained experience from involvement in other intervention projects, while others had limited professional experience outside their immediate workplace setting. For many participants, there was a certain level of excitement in being part of an intervention project, for example, some enjoyed working with modern technologies, while for others, it was a possibility to travel and meet colleagues or family. Some were wary about the prospect of a new system. They suspected foreign consultants would be insensitive to the local context and would present them with locally unintelligible solutions. Another suspicion was that the introduction of a new system would result in additional work instead of making work processes easier. Some users saw the task of entering data into the system as an end in itself, particularly if they already had experiences with other IS projects in which the reasons for introducing IS were unknown or not made clear to the users.

For Mozambican universities, the project provided a source of funding, e. g. of computer equipment. Universities were quick in spending these infrastructure investment funds. The relationship of universities towards the Ministry was one that can be characterized by a quote of a university manager: "Because we Mozambicans are poor, we are used to receiving from the Ministry. We do not normally contribute." This mindset is one of receiving financial and material items without reciprocating or engaging in a two-way process of joint problem solving.

8.5.2 Cross-cultural contradiction and conflict at the level of universities

The expectation of the Mozambican Ministry of Education and the Dutch project team was that the five participating universities would be sufficiently interested in the IS to contribute their own resources, at least modestly, in the implementation at their institutions. Opportunism to a varying degree on the side of the universities led to several conflicts. Even though all participating universities had committed themselves formally to the project, some local universities engaged in parallel, conflicting activities; soon after the project started, one of the universities decided to buy and implement an alternative, commercial system

from Brazil. Moreover, the attempt was made to convince the OPUS project partners to follow this example – which did not succeed. Another university repeatedly went into negotiations with commercial software providers but never decided on a product. In a third case, the university did not find the time and the energy to implement the system. However, at the remaining two universities, the system was, in fact, implemented and relevant aspects of this implementation are elaborated in subsequent subsections.

Another conflict arose from the fact that local software developers were trained extensively in the architecture of the system, but were never given the opportunity to put their expertise in practice. Their university was reluctant to invest its manpower into the project and preferred to allocate them to other pending tasks.

Furthermore, there was a desire for quick solutions, that is, the amount of effort within the universities was underestimated to make a new student information system work. Local universities saw the required effort primarily in technical installation plus participation in a few training sessions for the users. This led to long periods of inactivity within the institutions to prepare gradually for a working production system.

8.5.3 Reflexivity and change at the level of universities

The two universities that implemented the IS worked together to a certain extent during its implementation. At both universities, questions emerged when the system was put to use, many of which were brought to the attention of the broader project management team.

8.6 Inside the UCM

8.6.1 Structure and culture inside the UCM

The UCM was founded with the intention to provide higher education to the central and northern parts of the country, and therefore, its faculties are scattered over a large geographical area. In many respects, the faculties acted rather autonomously, and there was little centralized information management concerning student records. Only the mandatory statistics for the Ministry of Education were compiled centrally.

Academic registrars, as the principal users of the system, played an important role during the implementation of the IS. Each faculty had its own academic registrar. This was

considered a critical position since fraud would seriously undermine educational quality. To prevent fraud, predominantly Catholic sisters were put in these positions. The faculties varied in the student numbers, but one notable characteristic was that registrars used to know most students, meaning they could link names to the respective faces. With growing student numbers, this became increasingly difficult, but in the day-to-day work, it was remarkable to see how much student-related information they were able to retrieve from their memories, for example, about the study plans of students. This good memory often came in handy when academic data had to be gathered for certain purposes, such as the preparation of certificates or student profile sheets.

A characteristic of the work patterns of registrars was that data used to be gathered only at the time when needed. Some data used to be entered into Excel sheets, whereas other data was kept in printed form, for example, as paper student files and exam result sheets. Even though academic registrars were happy with the social aspects of being knowledgeable of students, most of them expressed their discontent about the shortcomings of the way information was managed.

Concerning the learning curve of users of the new IS, it could be observed that registrars tended to learn only enough to perform the task at hand. Instead of trying to get an understanding of the complete system, at first users only learned the most basic task of how to enter student records. At the end of the year, they learned how to move students according to their academic success, i. e. assign them to the subsequent study year or to repeat certain subjects in which they had failed. Some concepts were particularly difficult to learn, such as when curricula changed for new generations of students, and many users continued to live with an uncertainty about some aspects of the system. They would count on the availability of a qualified support person if they would encounter a difficult situation.

Technicians saw their role limited to technical system administration, such as the installation and technical maintenance, and avoided assisting users as far as domain knowledge was concerned.

External consultants were present at the UCM in different areas, such as in academic registry and accounting. Reactions by local counterparts varied when confronted with the perspective of externals arriving with new ideas concerning work practices, but often this was less of an issue when they realized that the foreigners would not be a threat to their positions, since the expatriates' presence was only temporary.

8.6.2 Cross-cultural contradiction and conflict inside the UCM

One contradiction was the centralization of student records into a single database. Before the OPUS project, details of academic information were not accessible to central university management. Although each user had its corresponding role in the IS, hence limited data access, a central database meant that the university management would potentially have access to all data, including data that used to be managed locally at faculty level. However, there was only limited actual conflict visible to date.

For academic registrars, the usability proved to be troublesome for certain tasks. This had its cause partly in the strongly database-oriented screen design mentioned earlier, and partly in the short implementation phase during the overall intervention project, which did not allow feedback from users to be integrated into the system's design.

Another contradiction was formed by opposing paradigms of incident-based data collection vs. timeliness of data entry. The IS was built based on the assumption that information is recorded at the time when it becomes available. In this way, reports and certificates could be extracted and printed based on correct data. But users were still used to incident-based data entry.

While users did not have technical knowledge, technicians did not enter the subject domain of academic registration but saw their mandate limited to technical system administration. This disjuncture had not yet led to conflict, due to the presence of external consultants who were – in their role as change agents – able to link the two areas, but the potential for future conflict exists.

8.6.3 Reflexivity and change inside the UCM

The intensive joint work between external consultants and academic registrars at the registrars' offices led to a better understanding of the value of well-organized, up-to-date information. The inherent limitations of local ways of administering student records were an additional motive for being open to other ways of working.

An important element for user acceptance was the possibility of integrating feedback from the users into system improvements. This was possible because of the availability of a small software development team at the UCM.

Central university management had been hesitant about creating a central unit for academic registry, because of the established autonomy of the faculties, but this changed

slightly over time because the new system still allowed decentralized working. Therefore, it was agreed to look for proper staff, both an academic registrar as well as an IS expert who could link the subject and the technical areas.

8.7 Discussion

In the following subsections, cross-cultural contradictions and conflicts will be discussed in terms of how different groups of actors applied different measures of success to certain aspects of the project. This serves as a basis to uncover the implications discussed in the following subsections.

8.7.1 Different measures of success

Structurational analysis of the project revealed a set of cross-cultural contradictions and conflicts, as summarized in table 8.1 (p. 197). The ITPOSMO model (Heeks, 2002b) is used here to analyze gaps in several dimensions between IS design and actual reality, i. e. differences between the actuality at IS introduction and the assumptions that guided IS design. According to the ITPOSMO model, the dimensions of relevance to design-actuality gaps are (1) *information* (data stores and flows); (2) *technology* (hardware and software); (3) *processes* (the activities of users and others); (4) *objectives and values*; (5) *staffing and skills*; (6) *management systems and structures*; and (7) *other resources* (such as time and money).

Table 8.2 lists the dimensions of design-actuality gaps related to previously identified contradictions and conflicts. Furthermore, the table shows related differences in success measures by different actors. Finally, corresponding success dimensions are listed. Different participants put different priorities on these dimensions. For example, the logic of intervention projects favored the delivery of agreed outcomes, which gave low priority to effective use. On the other hand, local organizations could only achieve impact if effective usage patterns were established. The dimensions are briefly elaborated in the following.

Local contributions Various issues indicated a disparity in the prioritization of local contributions to the project community. Local participants tended to classify themselves as *receivers* who waited for the delivery of artifacts and services, including extensive documentation. Such a delivery indicated success. On the contrary, the project coordinators

Table 8.2: Cross-cultural contradictions and conflicts linked to design-actuality gaps and differences in measures of success by project participants

Contradictions and conflicts	*Design-actuality gaps*	*Differences in success measures*	*Related success dimension*
Hesitation to 'get hands dirty'; Extent of documentation; Activities parallel to project	Objectives and values	Consuming artifacts and services *vs.* active participation	Local contributions
Transfer and ownership of source code; SW developers trained but unavailable;	Objectives and values; Staffing and skills	Exclusive ownership of resources *vs.* collaboration	Shared resources
Desire for quick solutions; Short implementation phase; Roles of users and technicians Centralization of student records Incident-based data collection	Management systems and structures; Other resources (time); Staffing and skills; Information; Processes	Accessibility *vs.* effective use	Embedding
Understanding requirements; Usability	Processes; Staffing and skills	Designers' understanding of requirements *vs.* actual requirements	Utility
Conflicting requirements	Technology	Separate solutions per organization *vs.* single adaptable solution	Scalability

associated success with certain levels of local engagement throughout the project to contribute to the system development and to gain local ownership. Moreover, parallel activities took place that were conflicting with the OPUS project, which undermined the involvement even further. These different positions represented gaps in objectives and values between both sides.

Shared resources One issue related to resources was the extent of ownership of project outcomes such as source code, and the willingness and ability to share resources with other universities. On the part of certain local participants, there was clearly a desire for exclusive ownership of project results, whereas the foreign project team envisioned sharing of the limited human resources of each of the local universities and wide reuse of project outcomes. Another issue was the ineffective human resource development; software developers of local universities were trained, but were soon after the training unavailable to any project related activities. Their new skills may have been useful for other activities, but there was clearly a conflict between the sponsors of the developer training and the unwillingness to make the developers available to project related activities.

Embedding This is the institutionalization of the information system in the work routines of the local universities. Locally there was a preference for quick solutions, ideally in the form of installing the system and immediately being able to use it without the need for lengthy configuration, training or tasks that involve collaborative action within the organization. This view puts a focus on *accessibility*: once the system is available, its advantages for the organization should be obvious enough for all necessary conditions within the organization to fall in place easily. In contrast, external experts considered it relevant to take advantage of the project time span to achieve *effective use*. This required to get all relevant organizational stakeholders, like users, developers and managers, more actively involved, in order for the local universities to create a sense of ownership, to gain experience and to actively come up with feedback about the progress of the system development.

However, effective use could not be established during project time. Organizational implementation did not receive enough focus. In addition to the problem of missing experimentation by universities, the logic of intervention projects did not contribute to better embed the OPUS system in the universities; external project management measured success of their own activities to a certain extent in the delivery of hardware, software and

training. This limited the focus of the overall project management on achieving effective use in the universities.

For the implementation of the IS, there was a discrepancy in what staff members, i. e. managers, users and technicians, saw in successfully executing their job roles, compared to what was necessary to achieve the intended organizational impact. In order to effectuate change, existing understanding of roles did not necessarily incorporate the required characteristics of the more interdisciplinary IS change agents.

Another issue was the concern of local participants regarding data confidentiality; traditional approaches were based on exclusive physical data access. This was not feasible anymore, because OPUS system is shared by multiple actors. Therefore, the exclusive access had to be traded for other techniques such as user authentication in a multi-user system environment. This implied assigning trust to technical system administrators.

Traditionally, the processes underlying information management by the registrars used to be incident-based, whereas the OPUS system design had timeliness was an implicit, unspoken assumption. This had an influence on information quality, because this change in behavior was not immediate. It was a routine that needed to be changed.

Utility The understanding of requirements by the designers was not always in sync with the reality. Several factors contributed to this disparity, such as the language barrier and the limited experience of local participants in formulating requirements. Also the usability was not always optimal, due to a close orientation of the screen designs towards the database data structure. This had an impact on the perceived utility of the system. For example, the OPUS system was not well prepared to work with large data sets, such as dealing with batches of student records.

Scalability Conflicting requirements among the different universities, such as distinct student number formats and report designs, presented a rather minor issue. Previously, most users had their own, home-made spreadsheets and documents, whereas the new information system was expected to be locally adaptable to specific settings. In this project, the issue of scalability could be resolved technically.

In order to achieve total success in all of the success dimensions listed in 8.2, local universities as well as external experts would have had to pursue numerous shifts in their (a) skills, (b) organizational structures, and (c) their views of what constitutes success.

Given such major challenges, it is comprehensible that the project was only partially able to achieve the intended goals.

8.8 Implications for IS in developing countries

In the following subsections, implications are discussed with respect to three perspectives: design and development, use and management.

8.8.1 Incremental development and frequent evaluation

According to structuration theory, structure cannot be inscribed or embedded in technology (Jones and Karsten, 2008). Nevertheless, technological properties, together with the institutional context and the power, knowledge and interests of human actors, do influence possible interpretations of technological artifacts by users (Orlikowski, 1992). However, interpretations of users are hardly predictable. Even so, the system development process needs to ensure the appropriateness of the produced technical artifacts.

In the project, system development followed an incremental and iterative approach (Brandon, 2006; van Vliet, 2008). Regular evaluation by future users of increments of the system being developed had several positive effects: First, iterative development helped to stay focused on relevant functionality and appropriate usability. Secondly, it kept local actors involved and allowed for mutual learning, thereby reducing design-actuality gaps.

This finding supports the iterative and incremental development approach that is characteristic of agile IS development methods (Larman and Basili, 2003).

8.8.2 Recurring use

Structuration is dynamic; social practices evolve over time and space. They must replicate even to stay the same, but often they evolve as they are reproduced (Rose and Scheepers, 2001). The interpretation of the IS by users follows the same logic. Baark and Heeks (1999) identified typical attempts to establish meaningful usage patterns, including user training, consulting and assimilation through day-to-day use. They considered day-to-day use as crucial, due to its longitudinal character. Over time, through repeated reinforcement by users, such practices become reified and institutionalized (Orlikowski, 2000). However, without ongoing user support towards the fruitful interpretation of the technology,

initial, suboptimal, usage patterns can congeal quickly and may even result in the rejection of the system.

Persistence over time in face-to-face sessions at the users' workplaces and the feedback of user experiences to system designers were important enablers for the institutionalization of effective information system use and the appropriate evolution of the technical artifacts.

8.8.3 Integrate design and use to leverage change

Users tend to view technology as closed, immutable systems when design and use are accomplished in separate organizations. However, even the most *black box* technology has to be apprehended and activated by human agency, and in such interaction, users shape technology and its effects (Orlikowski, 1992).

As already described, integration of user feedback helped to develop the IS's appropriateness-to-context and usefulness. An important enabling factor was the presence of a local change agent who brought together users and designers – backed by committed leadership. In the absence of a change agent, the danger was that the different participants tended to assume that "change management is the job of someone – or something – else" (Markus and Benjamin, 1997, p. 66). Change cannot be effectuated from a distance, but requires personal, longitudinal contact. At the UCM, this role was suitably taken over by external long-term consultants.

8.9 Chapter contributions

IS research in developing countries distinguishes itself in its attention to the local context of IS innovation and the developmental role of IS innovation (Avgerou, 2008). By considering several levels of analysis, this chapter has contributed to an understanding of how historical and social conditions of the participants in a cross-cultural setting have influenced the IS innovation process, and how IS innovation and culture mutually reconstituted each other over time through reflexivity and learning.

Historically constructed differences in, for example, attitudes to hierarchy and forms of communication have been found to affect participants' success measures along various dimensions. These success measures were not only diverse and conflicting, but

sometimes also volatile, and could even undermine initially agreed project objectives. Furthermore, contextual differences have been found to limit the involvement of future users, which enforced the time-space distance between consumers and producers and hampered flexible interpretation of the technology (Orlikowski, 1992).

Despite partial failure of the overall project, the analysis has shown conditions where structures in participants' minds were able to emerge that enabled successful IS production and use. Such an enabling environment was typically created by reflexivity and learning activities that involved the evolution of both technological properties and patterns of technology use (Orlikowski, 2000).

In practice, implications have been drawn based on the emergent nature of culture and the IS innovation. They concern IS production, use and management. The software development process requires rich feedback that reveals the multiple ways users interpret the IS in their institutional environment. Usage patterns can be nurtured by guided recurring use. To overcome design-reality gaps, a focus shall be put on change management.

8.10 Chapter conclusions

The chapter has attempted to analyze the process and the context of the cross-cultural IS development project targeting the Mozambican higher education environment. The analysis has included three levels, from global to inter-organizational to within the organization. It has led to insight about success being a multi-faceted issue. This is illustrated by the statement of the Mozambican national project coordinator that "compared to other projects this one has delivered good output." Furthermore, the system has attracted the interest from universities in other countries, as for example expressed empathically in an email conversation by an interested party from neighboring Zimbabwe: "[...] we are getting on well with OPUS and I feel it really fills a vacuum in the FOSS ecosystem insofar as SIS/SMS are concerned. I have used about half a dozen different FOSS SIS programmes and so far OPUS is the best fit for us."

In contrast, out of five local universities that participated in the project, only two have implemented the IS. Looking even closer at these two universities reveals that implementation has only been partially accomplished; only a subset of features, by a subset of faculties, are being used, mostly in parallel with traditional methods like Excel sheets. Additionally, future support for these universities has not been secured yet which puts a

question mark on sustainability. The contrast between the quote above and this short assessment indicates a wide variety of possible viewpoints of what constitutes success. The positive attitude of the manager can be related to experiences in other projects with less success, and to the future potential that he sees in implementing the software system in other Mozambican universities that suffer similar difficulties with respect to the administration of student records.

The chapter has drawn implications based on a discussion of diverse measures of success that were attributed to the IS by different project participants. The introduction of information systems always goes along with change, often regarding different dimensions. Such personal and organizational change needs to be managed, taking into account people's abilities. To accomplish change, persistence and committed leadership are instrumental. As information systems are developed and evolved, frequent evaluation of system increments facilitates relevance and usability, and provides opportunities for reflexivity and mutual learning. Both at project selection and during execution, it is beneficial to identify and focus on those local participants who are genuinely interested in the designed change that the IS intends to achieve.

Part III

Main result and conclusions

Chapter 9

Main result: Appropriate IS development (AISD) methodology

Previous chapters have considered a range of aspects that affect the successful and sustainable design and implementation of information systems in the context of developing countries. The findings of constructive work and evaluation in previous chapters are now condensed into an IS design theory, more specifically a methodology, called *Appropriate information system development (AISD)*.

The AISD methodology targets communities or organizations that have a need to implement an information system, but have little experience in IS production and use, and moreover suffer from weak resources, like limited availability of human resources and IT service providers. Their need can however not be met by an off-the-shelf system, but requires extensions to an existing system or the development of a new system. Because of the limited available resources, the organization is however not in a position to either develop the information system on its own in-house or to commission a local IS service company to do so. Inevitably, the organization needs to collaborate with partners to address its information system related needs. The partners may be geographically or culturally far apart, thus constituting cross-cultural projects. In many cases such partnerships follow the straightforward producer-consumer model, in which an IT service provider produces and installs technical artifacts, and eventually gives some user training. This model is however problematic, given the limited local IS experience and the large geographical and cultural distance between the user organization and the IT service provider. It leaves the

user organization left behind without the skills to technically and functionally maintain and improve the information system after the partnership ends. Hence, less experienced participants remain on a very low level on the scale of technological capability (see table 1.1, p. 17). Moreover, the analysis in chapter 8 has shown that in cross-cultural projects, local as well as remote partners may have differences in their success measures. Therefore, in order to achieve high levels of success, not only local skills and processes need to be established, but all participants may furthermore need to adapt their understanding of what constitutes success in order to form a shared understanding of success, based on two-way learning.

The proposed methodology considers both short-term and long-term success. It facilitates not only the production of an initial version of the technical artifacts (short-term), but also the learning of local partners on the way (long-term). Learning about information system development is time-consuming, therefore the knowledge exchange between partners with different IS experience is to occur during the course of the entire system development life cycle. Hence, the methodology's objective is two-fold: (a) create the IS product and (b) enable learning among the actors in the network. This addresses both short term interests, by creating a working IS, and long term interests by learning how to maintain and evolve the IS. This contributes to the ultimate goal of the AISD, to improve success and sustainability of cross cultural IS projects.

This chapter is organized as follows. The presentation of the Appropriate IS development methodology is based on the ontology for IS design theories formulated by Gregor and Jones (2007). The components of the design theory, which are summarized in table 3.6, p. 105, include: Purpose and scope (section 9.2); constructs (section 9.3); principles of form and function (section 9.4); artifact mutability (section 9.6); testable propositions (section 9.7); justificatory knowledge (section 9.8). After the description of the methodology, the same is put in relation to the research questions. This is done by formulating a set of theorems and arguing for their validity (sections 9.9 and 9.10).

9.1 Introduction to the methodology

IS design theories have as a primary design goal either a methodology or a product. This corresponds with the two meanings of the word *design* as a verb (to design) and as a noun (a design), respectively. For example, the system development life cycle (SDLC) is an IS design methodology, whereas the architecture and the functions of a word processor

would be a a product oriented design theory. In the present case, the interest lies with a *methodology*. The AISD methodology shall be capable of guiding cross-cultural IS production and use to greater success, with special emphasis on improving IS sustainability. The design theory is intended to be applicable to a wide variety of information system projects.

A methodology is a "recommended collection of phases, procedures, rules, techniques, tools, documentation, management, and training used to develop a system; we also note the importance of the philosophy behind the methodology, or the set of beliefs and assumptions underpinning it, explaining why the methodology functions as it does" (Avison and Fitzgerald, 2003, p. 80). For example, a methodology may be based on the belief that the key to successful development is the involvement of users throughout the IS development process. Beliefs and assumptions are often not explicitly stated by the authors of methodologies. To counteract this potential weakness, the application of the ontology outlined by Gregor and Jones (2007) is intended to make the methodology design as transparent as possible.

Chapter 4 has come up with two possible approaches to build an IS development methodology, which are inspired by the Appropriate ICT framework. At a high level, the Appropriate ICT is rooted in the system development life cycle (SDLC) and community informatics practices that put the local community or organization at the center. On the low level, tools and methods can be developed to guide the system development towards appropriateness. These two approaches are the basis for the AISD methodology. Another aspect of the Appropriate ICT framework is present in the AISD methodology: the distinction between product and the process of development. In AISD, the development process is strongly focused on learning.

The proposed AISD methodology respects the implications from the Mozambican OPUS project analysis in section 8.8 (p. 212): (a) users and developers need to engage in the habit of frequent evaluation of system increments in order to gain a shared understanding, (b) for users it is essential to get involved in a pattern of recurring use, in order to make the information system usage a matter of routine, and (c) to establish ongoing feedback between users and developers.

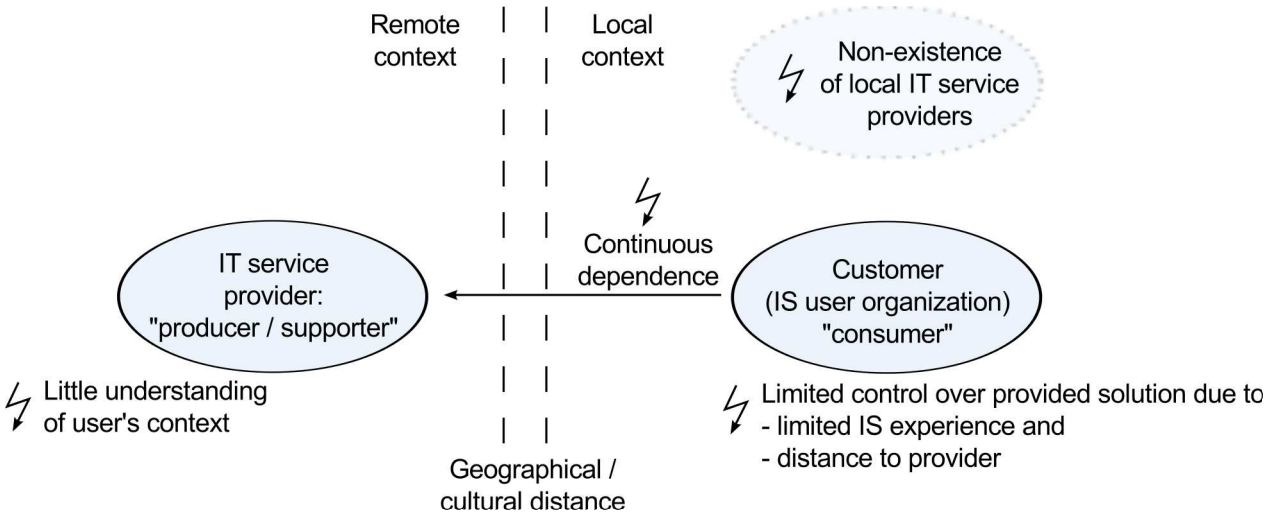

Figure 9.1: Problems for organizations in scarce resources contexts with little experience in IS projects

9.2 Purpose and scope

The problematic situation that is being addressed by the AISD methodology is illustrated in figure 9.1. When limited IS project experience and a scarce resource context come together, the widely practiced model relationship between customer (user) and IT service provider (producer, supporter) is confronted with several difficulties. First, in the context of scarce resources there may not be an IT service provider available that can meet the specific needs of the user organization. Collaborations with remote providers imply greater geographical and cultural differences. Second, organizations with little IS project experience have limited control over the provided solutions, since they are not experienced in stating requirements and evaluating solutions. The problem of limited control is intensified by the distance to the service provider. Third, the lack of IS production skills equates to continuous dependence on service providers. Fourth, the distant IT service provider has limited understanding of the user organization's context and is therefore unable to provide optimal solutions.

Table 9.1 provides an overview of the AISD methodology. The methodology's purpose is to sustainably resolve the IS needs of an organization that has limited IS project experience and suffers from scarce resource availability. Not only shall the short term interest be addressed by producing an information system, but the local organization shall

Table 9.1: Overview of the Appropriate IS development methodology

AISD purpose	Resolve the IS needs of an organization with little IS project experience, in scarce resource contexts, in a sustainable manner
AISD goals	Short term: A working information system at local organization Long term: Learning to maintain IS
AISD approach	Collaborate with other organizations to overcome knowledge gap Two-way learning; user organization: gain IS experience; support organization: understand local context
AISD characteristics	Aims for appropriateness during all phases of the SDLC Proposes a set of artifacts, i. e. tools and methods Encourages the development and testing of further appropriate tools and methods

have the minimal required capacity to maintain and to evolve the system. Thereby, the dependence on remote IT service providers is reduced.

The AISD methodology follows a concept of IS project success that covers both short term and long term success, thus explicitly including sustainability in the overall success. The structurational analysis in chapter 8 has identified five success dimensions: Local contributions, shared resources, embedding, utility, and scalability. The goal of IS sustainability is considered as part of overall IS project success. The aspect of sustainability can be broken down into three aspects that are part of the list of success dimensions, and which the method tries to meet: *utility* of the resulting system, *embedding* of the IS in the user organizations, and building of the necessary *resources*.

The AISD methodology intends to provide a flexible mechanism to nurture local capacity over time, while at the same time utilizing the given capacities from other actors in the network. A balancing effect of knowledge between the different participators is intended. The methodology shall facilitate the gradual elevation of skills so that initially less skilled actors can become active technology developers.

There is an intentional long-term view; the structurational analysis in chapter 8 has shown that a persistence over time is an important factor for IS project success. It is important to take into account that within the typical time spans of donor projects it may not be possible to carry out IS development projects in a sustainable way. That is,

the learning process may not achieve the ideal result, which would be sufficient capacity to local innovative production. However, donor funds can be vital for parts of IS projects, for example in early phases. The methodology needs to support external input in terms of financial and human resources, but shall not be dependent on it in the long term. One possible scenario would be a cooperation project that is funded in the initial time span, during which not only a useful technical artifact is being created, but also organizational structures are put in place and individuals undergo essential capacity building. Until the end of the externally funded cooperation, the achieved results would be convincing enough to make own funds available by those institutions that use the IS. External experts who have been involved may offer professional support as required to IS user institutions and first line support organizations.

9.3 Constructs

Constructs are the language of the design theory. The environment, in which phenomena of interest in IS research reside, consists of people (P), organizations (O) and existing or planned technologies (T). According to the structuring of the problem space into three domains, a combination of technology-based, organization-based and people-based constructs is part of the proposed design theory.

The basic constructs are shown in figure 9.2. Three types of constructs are distinguished: organizational constructs, people oriented constructs, and technological constructs. Organizational constructs include a network of user organizations, support organizations and project coordination entities. At this basic level of constructs there are no detailed assumptions made such as which form the project management organizations take, be it a distributed project management over various organizations or an open source foundation.

People related constructs comprise users, developers, managers, system administrators and consultants. These roles can be found in different organizational environments. Managers are associated with project leadership. Users include all those actors who have an interest in the system's functionality. This includes users who input data as well as managers who base their decisions on data provided by the information system. Consultants include those who offer support services to user organizations such as organizational IS implementation and adaptation. Developers and system administrators are technically oriented and may be based either directly in user organizations or in support organizations.

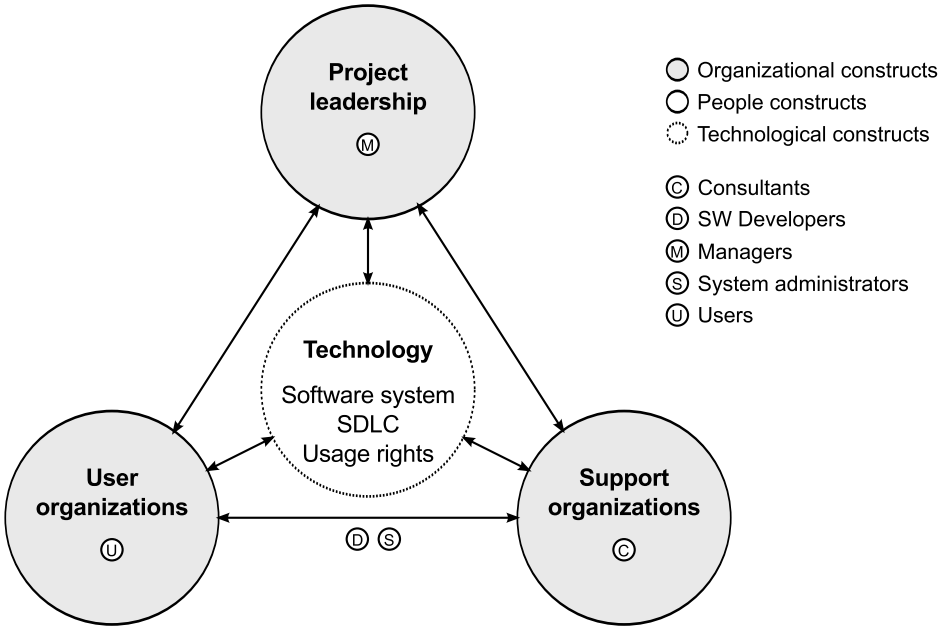

Figure 9.2: Basic organizational, people and technological constructs

Technological constructs include the software system, the system development life cycle (SDLC) and the copyrights and usage rights of the source code. The software system can be distinguished in the standardized software version, and localized SW versions that are adapted to specific user organizations. The SDLC provides a distinction of the life span of the software system into different phases, from definition to operation. The usage right of the source code is an issue to be sorted out because of the multiple contributing organizations and individuals, not least to establish community trust.

9.4 Principles of form and function

9.4.1 The AISD framework

Figure 9.3 is a representation of the AISD framework. The basic principles are the appropriate technology principles and the effective use orientation. Appropriate technology has a focus on available local resources and aims at increased local technological self-reliance and technological capability (Kahen, 1995; Tharakan, 2006). Effective use is a concept from community informatics and emphasizes active technology production by the local organization, in preference to mere access to technology, which is passive (Gurstein, 2003).

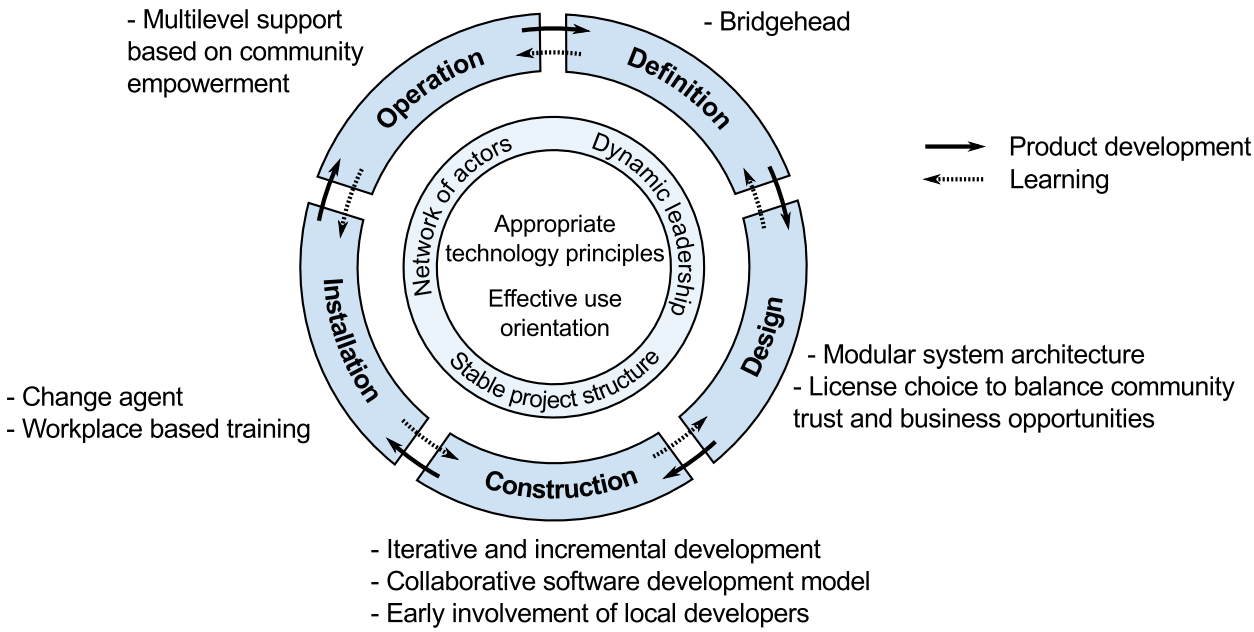

Figure 9.3: Appropriate information system development methodology

The system development life cycle (SDLC) is another pillar of the methodology. It follows the Appropriate ICT framework. The tools and methods are organized along the different phases of the SDLC. Certain artifacts are related primarily to a particular phase in the system development life cycle, others are relevant throughout the entire life cycle. Figure 9.3 indicates two different directions: One is the direction of product development, which cycles iteratively through all stages, from definition to operation. The opposite direction is about learning, as described in section 5.3.4, p. 153.

9.4.2 AISD tools and methods

Table 9.2 shows the tools and methods of the methodology. Some are process oriented, such as iterative and incremental development (IID). Others are structural, for example the stable project structure. Higher level constructs are tools and methods that work on top of the basic constructs.

Table 9.3 lists the AISD's suggested tools and methods and their contribution to the two goals of product development and learning. A working information system shall be built as quickly as possible. During the process of building, learning shall take place so that the local organizations can maintain the system in the long run. The table also indicates

Table 9.2: People-based, technology-based and organization-based artifacts as constructs of the IS design theory

Organization-based artifacts	A network of collaborating institutions with varying IS experience
	Dynamic leadership based on open source principles
	Stable project structure
	Multilevel support structures that aim for community empowerment
People-based artifacts	Change agent
	'Bridgehead'
	Workplace based training
	Early involvement of local developers
Technology-based artifacts	Iterative and incremental development method
	Modular system architecture
	Collaborative software development model
	Software license structure

to which success factor each product or learning item contributes. For example, the tool *support structure* enables a *feedback mechanism* from users to developers, which contributes to the success factor *utility*. The support structure also increases people's knowledge through training (success factor *resources*) and the institutionalization of the information system in the user organization (success factor *embedding*). The tools and methods are described in further detail in the following. It is shown how each tool or method contributes to product development and to learning, and which success factors are affected.

9.5 Tools and methods in relation to the overall methodology

In design science research, *design artifacts* refer to innovations that are built to address yet unresolved problems. They are evaluated against the usefulness that they provide to solve the problems (Hevner et al., 2004). Here, tools and methods are described that are part of the AISD methodology. For each tool or method the contribution to product development, to learning and to the success factors are described.

Table 9.3: Relationship of AISD tools and methods and the methodology's two goals of product development and learning

Tool or method	*Product*	*Learning*
Actor network	Joining resources (resources)	Balance knowledge (resources)
Dynamic leadership	Quick start towards utility (utility) Steering committee (resources)	Meritocracy (local contributions)
Stable project structure	Identification of roles (resources)	Human resources development (resources)
Support structure	Feedback mechanism (utility)	Empowerment oriented support (resources, embedding)
Change agent	Drive IS introduction (embedding)	Networking (resources)
Bridgehead	Understand local context (utility)	Facilitate feedback (utility)
Workplace based training	Feedback (utility)	Problem solving (embedding)
Early dev. involvement	Incremental contributions (utility)	Incremental learning (local contributions)
Iterative and incremental development	Frequent feedback (utility)	Evaluation of increments (local contributions)
Modularity	Minimize dependencies (scalability)	Take specific responsibility (local contributions)
Coll. SW dev.	Multiple sources (scalability)	Incremental learning (local contributions)
License	Resolve usage rights (scalability)	Access to source code (resources)

9.5.1 A network of collaborating institutions with varying IS experience

Product: Joining resources Because a single organization doesn't possess the required capacity to produce a satisfactory information system, a network of collaborating institutions is set up. This extends the set of resources available for IS development. The required skills to produce the system can be present at other nodes than the user organization – at least in the short term. The affected success factor is *shared resources*, mainly concerning the available skills, but it may also be extended to financial resources if e. g. donors are part of the actors in the network.

Learning: Balance knowledge The network not only provides resources, it is also a framework for knowledge sharing. More specifically, it enables two-way learning; user organizations can learn skills related to IS production and use, whereas support organizations can learn about the local context of the users. There may be considerable geographical and knowledge gaps between users and supporters, but this direction of learning may easily be overlooked. The two-way learning has a balancing effect of available knowledge in the network. This contributes to *shared resources*.

9.5.2 Dynamic leadership

Product: Quick start towards utility While overall leadership needs to be shared from the beginning between participants of the different contexts, a lot of early technical system development activities are carried out or supervised by the more experienced actors in the network. This is useful in order to quickly maximize the utility of the system for user organizations. In the case of open source software development, a possible scenario is a *spin-out* of a working software system (see table 5.2, p. 141). The quick start mechanism contributes to the success factor *utility*.

Learning: Meritocracy Leadership shall be flexible at two levels: first, responsibility for certain modules of the system, and secondly, leadership for the overall project. A variety of potential contributors shall be able and encouraged to take responsibility for certain modules of the system. Thereby, contributing actors can build up reputation and gain influence based on the quality of their contributions. The open source culture supports the maximization of reputation incentives by ensuring that peer credit goes where it is due and does not go

where it is not due (Raymond, 1998). Those who have built up strong reputation, shall also be enabled to gain influence on the overall project level. This meritocratic approach facilitates *local contributions*.

Product: Steering committee The overall project steering mechanism is concerned with the evolution of the system in a way that the different interests of user organizations are incorporated. Possible are inclusive structures such as open source foundations (see figure 5.2). This aspect contributes to the *scalability* of the project by enabling negotiations between the actors in the network.

9.5.3 Stable project structure

Product: Identification of roles The stable project structure provides an outline of important roles to implement projects with the AISD methodology. The stable project structure is illustrated in figure 6.1, p. 160). The different roles comprise (a) the management team, (b) the requirements team, (c) the operations team, (d) the development team, (e) the exploration team, and (f) the maintenance team. Not all roles need to be present at the user organization right from the beginning; certain tasks can be taken over from external partners until skilled people are in place locally to take responsibility. Skills are part of resources, hence the concerned success factor is *shared resources*.

Learning: Human resources development The stable project structure provides the set of required roles. The learning path, which runs from the operation phase towards the definition phase, indicates the sequence of learning; operations and maintenance teams are important in the early phases of the project to frequently install and test the system at the prospective user organization. Then, the development team gains importance in order to build local skills to participate in IS development activities, e. g. by designing report templates, adapting and translating the system. Later, the local requirements and exploration teams become important. Although requirements do have high priority already at the beginning, defining requirements is a task that requires highly skilled people. Therefore, typically an organization with little IS project experience is not readily capable of specifying requirements. Operation and development tasks are easier accessible to less experienced actors. The management team needs to exist from the outset, otherwise an organization would not embark on an IS project in the first place – even if the management team only

consists of a single person. This aspect concerns people's learning, therefore it belongs to the success dimension *shared resources*.

9.5.4 Multilevel support structures that aim for community empowerment

Product: Feedback mechanism For the implementation at user organizations and the continuous evolution of the IS artifacts, the input by users shall continuously be captured. Requirements tend to change over time. This requires ongoing improvements to the software system. Users need a point of contact to share experiences, including problems and ideas of IS usage. Because of the *stickyness* of information, distance between users and developers is detrimental to resolve issues. The distance should be kept minimal. Therefore, support is organized in multiple levels. Depending on the knowledge and capacities of the user organization in which an issue emerges, it may be resolved directly. Otherwise, a regional support organization is the next point of contact. Ultimately, support organizations from other regions or at the level of the central leadership are further options. The feedback mechanism contributes to the success dimension *utility*.

Learning: Empowerment oriented support Support organizations are oriented towards ongoing empowerment of user organizations. Some of the support services, such workplace based trainings, are described as separate artifacts, because they make important contributions to the overall methodology. This does not preclude them to be offered as part of support centers' services. Possible empowerment methods include (see table 7.3):

Service delivery: help desk, training, technical installation, development of additional functionality

Capacity building: face-to-face trainings at workplace, change agent consultancy, coordinating source code contributions

Advocacy: maintain a project website, maintain flow of information with user organizations (e. g. available updates) and central project coordination

Social mobilization: user group meetings, moderating on-line forums

Support centers need to be self-sustainable. Therefore, a business plan can be a useful exercise when setting up support organizations. The empowerment orientation aims to improve the relevance and attractiveness of support centers to user organizations. Support activities can aim at improving skills, for example the workplace based training,

or the embedding of the information system into the organization, like in the case of the change agent consultant. Empowerment activities primarily aim at strengthening *resources* and *embedding*. In some cases, the utility may be raised by development of additional functionality.

9.5.5 Change agent

Organizational practices are probably in place because they worked in the past. Changing them is one of the challenges during IS introduction. People have routinized certain practices. Additionally, managers, users and developers may have different intentions. It is not uncommon that users have hidden agendas when new information systems are introduced. This kind of obstacles needs to be dealt with. One commonly observed danger is that organizational change is thought to be "the job of someone – or something – else" (Markus and Benjamin, 1997, p. 66); often the power of change is attributed to the technical IS artifact itself, that is, without further intervention the new system is expected to be attractive enough to encourage users to change their practices accordingly. This is hardly a realistic scenario. In such a complex environment, change agents try to make change happen, either as more neutral change facilitators, or as more political change advocates (Markus and Benjamin, 1997).

Change agents are useful during the entire IS development project. Figure 6.3 (p. 171) shows possible scenarios of change agents in IS user organizations. Change agents are ideally placed within user organizations, since they play a decisive role in smoothing the way and to take advantage of new IS based opportunities. Because in many cases (prospective) IS user organizations may be unable to identify suitable change agent, external consultants may be hired to take over the role.

Product: Drive IS introduction The change agent prepares the ground within the organization concerning skills and technology. He brings people together to make the necessary decisions so that the IS introduction is as successful as possible. The primary affected success factor is *embedding*.

Learning: Networking The change agent brings together people from all kinds of roles as necessary in order to drive the IS introduction. This also leads to knowledge exchange between people, contributing to skills, i. e. *shared resources*.

9.5.6 'Bridgehead'

Product: Understand local context To reduce cultural gaps between actors in globally distributed software development projects, the concept of *bridgehead* has been applied with success. This refers to the presence of one or more experienced individuals at IS user organizations to facilitate mutual understanding. This can include overcoming language barriers if different actors speak different languages. But it also means to better understand requirements. Requirements are often hard to express by users even in face-to-face situations, and this is typically made worse with cultural distance. Bridgehead teams can lower both organizational and national culture gaps (Carmel and Agarwal, 2001). Understanding the local requirements better is a valuable input towards *utility*.

Learning: Feedback The bridgehead who is present at the user organization has many possibilities to bring knowledge about the product to the user organization. For example, the bridgehead can do workplace based learning sessions to train the users, or pair programming sessions to train the local developers. In order to strengthen the connection between the local and the remote sides, the bridgehead is able to set up a feedback mechanism, so that the user organization overcomes possible difficulties in talking to other organizations in the network. The feedback mechanism can provide important information about bugs, feature requests and other aspects of the local installation. Thereby it contributes to *utility* of the standardized version of the system.

9.5.7 Workplace based training

Learning: Problem solving Work place based training sessions tackle real-world problem situations as they emerge naturally. It is an effective technique to help institutionalization of new information systems. Such workplace-based sessions take considerable time. It is not sufficient if they remain single events. They need to be repeated over a certain time period until new usage patterns are established. Repeated reinforcement of the newly learned practices makes it possible to change old ways of working to new, more effective ways (Orlikowski, 2000). Instead of reproducing old patterns, new ways of understanding and working need to be created. To achieve this, face-to-face sessions at the workplace are useful for mind shifts that create new mental structures (Giddens, 1984). They contribute to the *embedding* of the information system.

Product: Feedback For product development, work place based sessions provide immensely rich data that can be used to improve the system. Not only do the users learn how to use the system in certain real world situation, but also suboptimal characteristics of the system are highlighted. This includes bugs, but also ineffective, clumsy user interfaces. This contributes to the system's *utility*.

9.5.8 Early involvement of local developers

Learning: Incremental learning Learning of local developers runs in the reverse direction of the SDLC, from learning how to operate to learning how to define desired information systems (see figure 5.3, p. 155). Learning takes time. Therefore, despite an early lead by the networks' more experienced developers, local contributions are desired right from the start of the project. Less experienced developers may receive some additional training as required, but an essential component for their capacity building is to take over increasingly demanding tasks in IS development. Since all participants are able to work on a joint source code repository, it is possible to make contributions by developers located anywhere as long as basic Internet access is available. This learning aspect contributes to *local contributions*.

Product: Incremental contributions Local developers naturally have a better understanding of the local context and can therefore better serve the given needs of a user organization. By contributing right from the beginning of the project, they can improve the *utility* of the system.

9.5.9 Iterative and incremental development method

Users play an important role in any IS development. Iterative and incremental development (IID) is equal to feedback-driven refinement with customer involvement and clearly delineated iterations (Larman and Basili, 2003). The iterations are typically in the dimension of weeks, but can also be as small as days. Agile, IID-based, practices are relevant both in project management (e. g. Scrum) or developer oriented practices (e. g. Extreme programming).

IID methods have been used successfully in globally distributed software development projects. For example, Scrum simple planning techniques for coordination of tasks have been shown to increase 'teamness' and reduce the felt geographical distance between

users and developers. Additionally, XP practices such as testing and refactoring have helped to maintain a high quality of the source code (Holmström et al., 2006).

Product: Frequent feedback Thus, IID can have a positive impact on maintaining system relevance during development. Frequent testing and evaluation by future users provides valuable feedback for the IS design. The IID approach is also intended to take optimum advantage of the limited times when participants from distant locations are able to come together and review the project progress. In practical terms, such meetings are typically associated with considerable travel costs. At times when participants cannot meet physically, or between such meetings, iterative and incremental development still is a way to reduce distance between the actors. Frequent feedback improves the *utility*.

Learning: Evaluation of increments The iteratively and incrementally growing system is an ideal 'playground' for its frequent evaluation by future users. When users evaluate, they inevitably learn important aspects of how to use the system. Their evaluation feedback is a *local contribution* to the IS development process.

9.5.10 Modular system architecture

Product: Minimize dependencies Breaking down large code bases into smaller units is a long recognized technique (Parnas, 1972). It supports sustainable software development by eliminating excessive dependencies between modules, which is one of the common causes for soaring costs of software system change (Tate, 2006). The guiding principle for decomposition is information hiding, i. e. putting together sets of information that rarely need to cross module boundaries. Modularity helps to design a *scalable* system; not all the existing functionality may be desired by every user organization. It is also possible to have alternative modules for similar functionality, from which user organizations can choose.

The integration of modules requires some sort of extension mechanism. Otherwise there is a risk of introducing dependencies between modules that have a negative effect on the maintainability of the code. An extension mechanism makes it possible for each module to dynamically define extension points, which can be implemented by other modules. This facilitates work coordination between different contributing teams or individuals. It also enables user organizations to adapt the system with organization-specific functionality. An

example of an extension mechanism is OSGi (`http://www.osgi.org`), which is used for example in the popular Eclipse platform (`http://eclipse.org/`) (Aßmann, n.d.).

Learning: Take specific responsibility The organization of the overall system in smaller, functional units, makes it possible for different actors to take over responsibility for modules. Hence, local organizations can develop required functionality in a separate module, and be responsible for the module. This makes it easier for *local contributions* to emerge.

9.5.11 Collaborative software development model

Product: Multiple sources The strength of a collaborative model lies in the extent to which forces from multiple sources can be joined by the partners within the network. Contributions need to accumulate so that a more suitable solution can be created that would not be possible by any of the actors on its own. Figure 5.2 (p. 148) outlines how various partners can collaborate to create a standardized core system and localized versions that adapt to the user organizations. To get up and running relatively quickly, seed funding can provide a basis on which to expand. User organizations would ideally have developers, but also those with only users shall contribute to the software development process through feedback. The lead development initially comes from more experienced partners. Over time, more and more of the technical tasks are taken over by local developers. A setup that allows integrating contributions from multiple sources facilitates a *scalable* network.

Learning: Incremental learning A collaborative software development model makes participation of local participants possible. Therefore it is an important building block to enable local learning based on *local contributions*.

9.5.12 Software license structure

Product: Resolve usage rights The license structure has vital importance for the *scaling* of the IS. It deals with the resolution of ownership issues that can otherwise consume a lot of effort, especially when the number of actors grows. Open source licenses are a possible license choice to coordinate collaborative action. A proper open source license choice can strike a balance between community trust and business opportunities. In such a scenario, the core module would typically be put under the open source license, whereas

other modules may be either published under open source or commercial licenses. If other licenses than open source licenses are used, an essential point to consider is how to resolve ownership issues concerning the code and documentation that is being produced by different actors.

Learning: Access to source code The license clarifies the accessibility to the source code. In case of the application of an open source license, the access is ensured for all interested parties. Source code becomes a *shared resource*. This creates possibilities for learning, because the source code can be studied.

9.6 Artifact mutability

The AISD methodology comes with a proposed set of tools and methods that contribute to the five success dimensions, namely local contributions, shared resources, embedding, utility and scalability. The set of tools and methods is however not fixed, but further artifacts can be developed, used and evaluated.

The methodology is meant for actor networks with a flexible set of actors. It may start as collaboration between multiple actors from the outset, or it may start with an isolated pilot project and later scale. Over time, further organizations may become interested and join in. The process is an ongoing balancing act between generalization and contextualization, as illustrated in figure 2.3 (page 60). Standardized and localized versions of the system need to be maintained.

The designed software is not a static artifact, but can be improved and adapted by any of the participating institutions. A distinction has to be made between merely locally relevant changes, e. g. localizations to a particular setting, and changes that are of sufficient quality and of interest to other user organizations. In the case of changes that are only of local relevance, the extension mechanism can be useful in order to preserve update compatibility as far as possible. In the case of improvements that are of wider interest, the changes can be incorporated in the central source code repository.

The level of input by individual actors is also flexible. Typically, in the early stages a lot of contributions are made by the more experienced participants. Later this can shift towards other nodes in the network. This opens up the possibility to withdraw donor support at the time that other actors have reached a situation of being self-sufficient in

maintaining support activities.

Similarly, leadership is dynamic in that the responsibility for maintaining the source code can change between participants. Leadership is a decisive component of open source software development, and participants that have made important contributions to the project are potential candidates to take over responsibility for certain modules. The dynamic leadership approach based on open source principles intends to give responsibility to those that have contributed extensively and demonstrated a mature understanding of the system development activities.

For coordination and to be able to vote on strategic decisions, once the open source network has reached a certain size, a central coordinating body, such as a project foundation, may offer a forum to bring together representatives of the different actors in the network.

An important point related to the sustainability of an IS initiative is its potential to scale from pilot settings to a larger set of actors. For example, financial sustainability of a regional support center is only possible with a crucial minimum set of user organizations that act as clients, in demand of support services. A support organization may however emerge from support services offered by an internal unit at a local university.

9.7 Testable propositions

What the methodology promises is to improve the success and sustainability of IS initiatives, particularly when actors in less experienced contexts are involved. The methodology is targeted to take advantage of existing capacities in the network and transfer them to other actors who are in demand of those capacities. Therefore, testable propositions need to address IS sustainability.

- The methodology facilitates local capacity building through knowledge transfer between different nodes in the actor network.
- Given a critical amount of user organizations in a particular region, support structures can be established that are both financially viable and able to deal with emerging user requests.
- Given the availability of sufficient knowledge in the network as a whole – not necessarily in each organization – information systems can be designed that are sufficiently useful for user organizations.

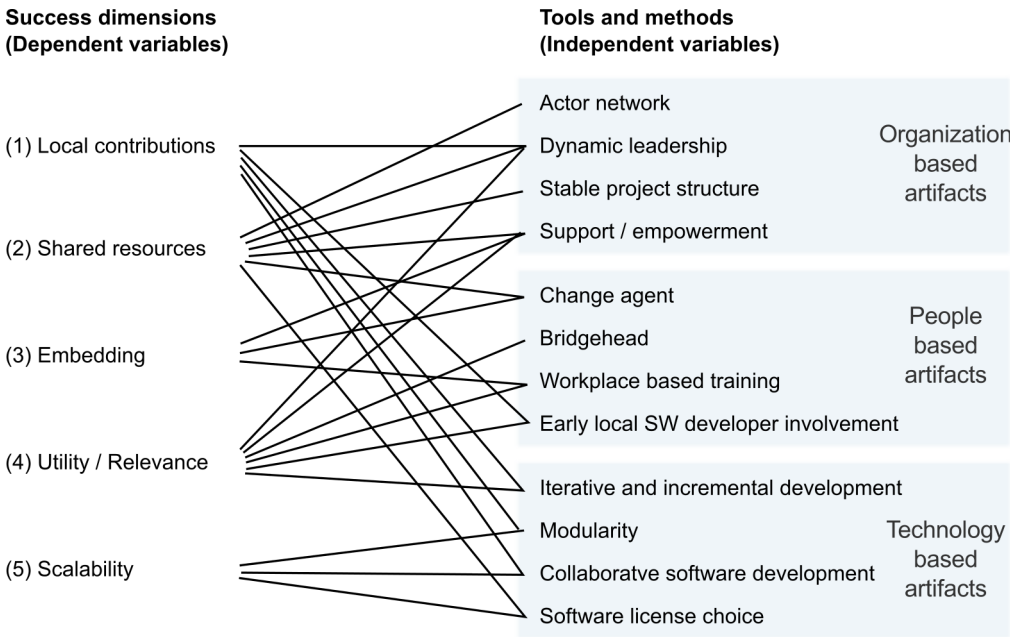

Figure 9.4: Proposed relationship between IS design theory artifacts (tools and methods) and success dimensions

- A sufficient level of embedding of the IS into user organizations is feasible for organizations with different levels of technical capabilities, depending on the quality of the regional support center services.
- The scaling of actor networks is possible.
- Input from actors can vary. The withdrawal of donors or other actors with initially high levels of capacity is possible after other nodes have established sufficient capacities and institutionalization.

Figure 9.4 displays relationships between success dimensions and IS design theory artifacts. The success dimensions as the dependent variables are thereby influenced by the application of the artifacts that are part of the design theory. These relationships are proposed on the basis of project experience and are candidates for further testing in related research.

9.8 Justificatory knowledge

The validity of a design theory improves with a thorough grounding in three different processes (Goldkuhl, 2004): empirical grounding, theoretical grounding and internal

grounding. These three aspects are elaborated with respect to the AISD methodology.

9.8.1 Empirical grounding

This is grounding in observations and the practical application of the design theory. The longitudinal action research project provides the practical grounding for the design theory. In particular, the structurational analysis in chapter 8 has analyzed many of the concepts that have been developed and put into practice during the action research project; whereas previous stages have focused on particular concepts (e. g. appropriate technology or open source), the structurational analysis took a global perspective of the complete project.

9.8.2 Theoretical grounding

This is grounding in external theories. A variety of theories were drawn upon throughout the chapters of this thesis. They have been integrated into the AISD methodology. The methodology is based on theoretical foundations from several fields:

- Sustainability with respect to various fields
- Information systems success and sustainability
- Appropriate technology, Appropriate ICT
- Free and open source software
- Software development methodologies
- Community informatics: effective use
- Community empowerment
- User-oriented innovation
- Evaluation based on structuration theory

The objective of the AISD methodology is to improve the success of cross-cultural information system projects. In this respect, local appropriateness and sustainability as an element of success play a particularly crucial role. Appropriate technology has a long tradition in proper technology selection, design and maintenance in diverse contexts. Interest in appropriate technology is now emerging for ICT projects. Therefore, appropriate technology has been chosen as an important theoretical base, to be applied to IS production and use.

Sustainability is a concept that is primarily viewed concerning its environmental roots, but is nowadays being applied to a variety of disciplines. Here, sustainability has been

considered ranging from context-free sustainability concepts to IS specific sustainability.

Free and open source software development is now an established method for the production of software. It minimizes the barriers for actors across national boundaries to get involved in the active production. This aspect of active local involvement is also emphasized by community informatics with the effective use concept. Not merely access to technology counts, but also the shaping of one's own future by engaging actively in order to solve local problems. Open source is conducive to achieve effective use, because it facilitates user-oriented innovation.

A variety of software development methods emerged over the last decades. The choice of software development methods can have strong impact on issues such as collaborator coordination and IS relevance. Modern, agile approaches are able to improve globally distributed development projects. Therefore, they form another theoretical basis of the AISD methodology.

9.8.3 Internal grounding

This includes conceptual grounding and value grounding. Values are linked to the intended goals that are supposed to be achieved. This includes the stated purpose and scope that has been laid out in the introduction of the thesis, and summarized as the purpose of the theory in section 9.2. For example, this thesis – and consequently the methodology presented in this chapter – are based on the belief that ways to tackle underdevelopment should aim to put people in control of their future and therefore make them active participants of the production of technology. It is also believed that many current approaches do not adequately address this issue.

Conceptual grounding is linked to the fact that all statements, including prescriptive, explanatory and value statements, include the use of categories. Therefore, it should be made clear, which categories are used in a design theory. Conceptual grounding means to outline the ontological basis for the prescribed action by the design theory. In this chapter, the ontology by Gregor and Jones (2007) was used to describe the constructs and the principles to put the constructs to work in practice.

9.9 Properties of AISD

The research in this thesis was guided by a set of research questions (see section 1.5). In order to demonstrate that the AISD methodology corresponds to the stated research questions, a set of theorems are proposed here. The validity of the theorems will be justified by reasoning.

Research question 1. How can information system development projects deliver meaningful solutions to local problems? In other words, how can the relevance of information system development projects be ensured?

Theorem 1. *Appropriateness: AISD is useful to develop locally relevant IS solutions in contexts of weak infrastructure.*

The argument for this theorem is based on the rationale that solutions need to fit the local context. It is often believed that theories, models and technology that are developed in industrialized countries are universal, but experience in the developing world shows that this is far from true. The AISD methodology is based on appropriate technology principles and builds upon the Appropriate ICT framework. Appropriate technology principles include the use of available means to satisfy local needs (Tharakan, 2006). AISD does that by augmenting the available means through collaboration. AT aims to achieve greater technological self-reliance and technological capability, together with the fulfillment of developmental goals (Kahen, 1995). It tries not only to use existing, but also to build additional skills and resources to raise the local productive capacity (Akubue, 2000). AISD aspires skill building and increased self-reliance through its learning cycle, while at the same time achieving system development goals through the product development cycle. AT was envisioned as technology that is immensely more productive than existing technology, but at the same time much more affordable than sophisticated, highly-capital intensive technology of industrialized countries (McRobie, 1979; Schumacher, 1973). AISD provides an alternative path that is a middle-ground between two extremes. On the one side are the existing, inefficient and error prone information system technology like pencil and paper or Excel. On the other side are unaffordable, highly sophisticated technologies of the likes of SAP or Oracle, which often come with expensive consultants.

AISD suggests a set of appropriate organization-based, people-based and technology-based artifacts to guide IS projects. These tools and methods are oriented towards appropri-

ate technology principles. This set of artifacts is not exclusive, but the AISD framework is open to the inclusion of further tools and methods. Some of the proposed tools and methods aim to improve the relevance, or utility, of the information system. An important way to facilitate relevance is the establishment of a feedback mechanism from users to developers. The bridgehead, iterative and incremental development, and workplace based training are tools and methods that establish feedback. The bridgehead facilitates understanding between two culturally distant partners and helps to better understand local needs. Iterative and incremental development proposes frequent evaluation of the incrementally growing system. This is a way to discover potential inappropriate solutions as early as possible, and to be able to react accordingly. The workplace based training is a real-world evaluation of the solution and facilitates the user-developer communication by providing feedback about shortcomings and other experiences of the system. Furthermore, the early involvement of local developers is a way to put local participants in a better position to independently provide solutions to local needs.

Research question 2. How can information systems be developed cooperatively, given the globally distributed character of the participants?

Theorem 2. *Collaboration: AISD enables collaborative IS development across cultural borders.*

The collaborative software development model is at the heart of the AISD methodology. This is a coordinating concept to organize the collaboration in the actor network. In open source development, the joining of contributions by different actors is a proved practice. Concerning the methodology's short-term, product oriented goal, a spin-out model is suggested to produce a basic working information system. Thereby the more experienced developers quickly develop initial versions of the desired system. On the way, dynamic leadership shall facilitate the shifting of contributions and responsibility from external to local participants, so that in the long term the local participants have the capacity to maintain and evolve the system. The collaboration is depicted in figure 5.2 on p. 148. The figure shows the different collaborating actors and also shows a possible involvement of a project sponsor.

In addition to the collaborative software development framework, several AISD tools and methods facilitate the collaboration: The bridgehead contributes to mutual un-

derstanding concerning the context and the requirements. A modular system architecture eases distributed development by making it possible for distributed teams to take ownership and responsibility of different functional modules. Not least, a proper license choice is essential to organize copyright and usage right issues. Putting source code and documentation under a license avoids the need to make mutual agreements between actors concerning usage rights.

Research question 3. How can local innovation be nurtured, so that local participants take active ownership and control in shaping solutions to local problems, rather than waiting for solutions to be delivered from outside?

Theorem 3. *Change: AISD facilitates organizational change and institutionalization during IS implementation.*

This research question and its corresponding theorem are related to the inside of the user organization. It concerns the integration of the newly introduced information system in the organization and making its use a matter of routine. Several AISD tools and methods are specifically concerned with the IS embedding. The stable project structure indicates the required roles for the long-term operation of the system. The change agent does what it takes to bring together people and technology to make the system work and to facilitate required organizational skills development. The early involvement of the organization's local developers contributes to the skills development and makes the organization more independent from partners. Iterative and incremental development makes it possible for the organization to be constantly involved in the evaluation of the information system as it is being developed. Thereby the organization can work on the embedding right from the beginning of the project.

Research question 4. How can information systems be supported in the long term, so that installed systems do not get abandoned despite their potential to solve local problems?

Theorem 4. *Support: AISD facilitates the creation of community based support structures.*

The envisioned support structure is guided by the principle that resolution to local problems shall be resolved as close to the source as possible. At the source of an occurring problem, information about the conditions is for free. The distance between the source of

the problem and its resolution increases the possibility of misunderstandings. Therefore, support services are targeted to empower user organizations to create a certain level of self-help competence. Support services have to be nurtured by the more experienced participants in the network, but over time regional support providers shall be put in place, in order to be closer to the user organizations. Support services are based on community empowerment at four levels: (a) Service delivery, which typically targets the improvement of the utility of the system; (b) capacity building, which improves relevant skill; (c) advocacy, which is about making interests heard of the different stakeholders in the network; and (d) social mobilization, which is about furthering the interaction within the network, but also within individual user organizations to institutionalize the information system. For example, the change agent contributes to social mobilization within the organization.

Research question 5. What are the relevant indicators of success and sustainability in cross-cultural information system project settings?

Theorem 5. *Critical success factors in cross-cultural information system projects include: Local contributions, shared resources, embedding, utility, and scalability.*

The set of five success factors have been identified in a case study covering the Mozambican OPUS project. These factors are based on the analysis of potential and actual conflict between all involved participants. The analysis was done by considering three levels: the international level, which covers the foreign experts and the Mozambican project management; the levels of the universities as the set of user organizations; and the level of one particular university, the UCM, which focused on individual users and groups of users. The analysis found that different participants had sometimes different ideas about what constitutes success. This means that different participants worked towards diverging goals. The set of five success factors is related to the minimization of conflict. Correspondingly, the AISD methodology aims to minimize conflicts by working towards the success measures.

9.10 The main theorem

Overall research question. How can the success and sustainability of cross-cultural information system development projects be improved?

Main theorem. *AISD leads to a successful and sustainable IS development environment.*

The main theorem is a direct consequence of the properties described in the previous section. It does not only guide technical information system development, but it intends to make the solution appropriate to the local context, by emphasizing learning along the way; it provides a model to tap shared resources by forming a network of collaborating institutions; it strengthens internal change and institutionalization; and it suggests community empowerment based support services, which have the potential to be economically feasible. Furthermore, it contributes to critical success factors, as illustrated in figure 9.4, p. 239.

Chapter 10

Conclusions

The research has covered a conglomeration of aspects that were considered relevant for the design and implementation of an information system in a cross-cultural setting. The practical case that was the basis for the investigation of the research questions was the OPUS north-south cooperation project between universities in Mozambique and the Netherlands. The research project was carried out primarily as an action research study. This did not only produce theoretical research results, but enabled also the finding of solutions to practical problems by project participants and researchers.

The research in this thesis involved several steps. First, the applicability of appropriate technology to software development and cross-cultural information system development was investigated. Next, an open source model was developed and applied to the OPUS system. Then, organizational issues related to change management and local innovation were investigated. Afterwards, a support structure was designed to provide a helping hand in case of user problems. The project as a whole was evaluated with a sociologically based approach, the structurational analysis, which was presented by Walsham (2002) as an evaluation tool for cross-cultural information system production and use.

Finally, the insights gained throughout the research process were put into the framework of a methodology, called the appropriate information system development (AISD) methodology. The methodology represents a theory for design and action, one of five types of information system theories (Gregor, 2006). The research in this thesis has tried to avoid one of the two extremes of either overemphasizing or black-boxing technology. The AISD methodology integrates socially oriented insights as well as technical aspects (e. g. open source, iterative and incremental development). In response to the research questions, the-

orems were proposed. It was argued how the AISD methodology corresponds with the theorems.

The research was based on a particular practical project. It cannot be claimed that the findings will necessarily be true for other information system projects. Nevertheless, generalizations are possible – not statistically to greater populations (Yin, 2009), but to (a) the development of concepts, (b) generation of theory, (c) drawing of specific implications, or (d) contribution of rich insight (Walsham, 1995b). All of these four possible generalizations were possible and have been carried out in this thesis. Its main contribution lies possibly in the generation of a design theory, i. e. a methodology for appropriate IS development, because it integrates concepts, implications and insights from other chapters.

10.1 Further research

One feature of IS design theories is that parts of a theory may deal with subsystems, which in turn can have their own separate design theory (Gregor and Jones, 2007). The tools and methods used in the AISD methodology (see chapter 9) have been the outcome of previous chapters. Some could be improved and sharpened by formulating dedicated design theories. For example, in section 4.5.6 there was an early attempt towards a more detailed and structured description of a tool to involve local software developers. It could be improved by formulating a separate design theory that specifies the involvement of local software developers throughout the IS project in greater detail. Although the AISD methods and tools contribute to all five success dimensions that were identified in the project analysis in chapter 8, it cannot be concluded that the set of proposed tools and methods is sufficient to achieve success in other IS development projects. Further tools and methods may be appropriate, depending on the context and on the background of the participants. For example, the round table process may be a valuable addition to the set of tools and methods. The round table process wasn't known to the project participants in the OPUS project, but has been shown to have relevance in ICT projects in the developing country context (Moens, 2010). Therefore, it could be investigated how the round table process fits with appropriate IS development. Furthermore, the AISD methodology has put forward testable propositions. Further research is suggested in order to verify them.

Bibliography

Abrahamsson, Pekka et al. (2002). "Agile software development methods: Review and analysis." In: *Vtt Publications* 478, pp. 1–107. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.161.5931&rep=rep1&type=pdf (visited on Jan. 26, 2012).

Agha, Syed Salim (1992). *Sustainability of Information Systems in Developing Countries: An appraisal and suggested courses of action*. Ottawa, ON: IDRC.

Ahmed, Allam (2007). "Open Access Towards Bridging the Digital Divide–Policies and Strategies for Developing Countries." In: *Information Technology for Development* 13.4, pp. 337–361.

Akubue, Anthony (2000). "Appropriate Technology for Socioeconomic Development in Third World Countries." In: *The Journal of Technology Studies* 26.1.

Alkhatib, Jamil, Mohab Anis, and Hamid Noori (2008). "Open Source: The Next Big Thing in Technology Transfer to Developing Nations." In: *International Association for Management of Technology IAMOT 2008 Proceedings*.

Avgerou, Chrisanthi (1995). "Transferability of information technology and organisational practices." In: *IFIP 9.4 Conference*. Cairo, Egypt. URL: http://eprints.lse.ac.uk/3594/ (visited on Mar. 5, 2012).

— (2000a). "Information systems: what sort of science is it?" In: *Omega* 28, pp. 567–579.

— (2000b). *IT and organizational change: an institutionalist perspective*. LSE research online. URL: http://eprints.lse.ac.uk/2582 (visited on Oct. 20, 2011).

— (2001). "The significance of context in information systems and organizational change." In: *Information Systems Journal* 11.1, pp. 43–64. DOI: http://dx.doi.org/10.1046/j.1365-2575.2001.00095.x.

Avgerou, Chrisanthi (2008). "Information systems in developing countries: a critical research review." In: *JIT* 23.3, pp. 133–146. URL: http://dx.doi.org/10.1057/palgrave.jit.2000136.

Avison, David, Richard Baskerville, and Michael Myers (2001). "Controlling action research projects." In: *Information Technology & People* 14.1, pp. 28–45.

Avison, David et al. (1999). "Action Research." In: *Communications of the ACM* 42.1, pp. 94–97.

Avison, David E. and Guy Fitzgerald (2003). "Where Now for Development Methodologies?" In: *Communications of the ACM* 46.1, pp. 79–82.

Aßmann, Uwe (n.d.). *Eclipse and Framework Extension Languages*. TU Dresden. URL: http://st.inf.tu-dresden.de/Lehre/WS06-07/dpf/slides/11-eclipse-2p.pdf (visited on Feb. 1, 2012).

Baark, Erick and Richard Heeks (1999). "Donor-funded information technology transfer projects: evaluating the life-cycle approach in four Chinese science and technology projects." In: *Information Technology for Development* 8.4, pp. 185–197. DOI: http://dx.doi.org/10.1080/02681102.1999.9525309. URL: http://dx.doi.org/10.1080/02681102.1999.9525309.

Bada, Abiodun O. (2002). "Local Adaptations to Global Trends: A Study of an IT-Based Organizational Change Program in a Nigerian Bank." In: *The Information Society* 18, pp. 77–86.

Bailur, Savita (2006). "Using Stakeholder Theory to Analyze Telecenter Projects." In: *Information Technologies and International Development* 3.3, pp. 61–80.

Baldwin, Carliss Y. and Kim B. Clark (2006). "The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model?" In: *Management Science* 52.7, pp. 1116–1127.

Bamberger, Michael (1991). "The politics of evaluation in developing countries." In: *Evaluation and Program Planning* 14.4, pp. 325 –339. ISSN: 0149-7189. DOI: DOI:10.1016/0149-7189(91)90015-9. URL: http://www.sciencedirect.com/science/article/pii/0149718991900159.

Barrett, Michael, Sundeep Sahay, and Geoff Walsham (2001). "Information Technology and Social Transformation: GIS for Forestry Management in India." In: *The Information Society* 17, pp. 5–20.

Baskerville, Richard and Michael D. Myers (2004). "Special Issue on Action Research in Information Systems: Making IS Research Relevant to Practice – Foreword Vol. 28 No. 3, pp. /September." en. In: *MIS Quarterly* 28.3, pp. 329–335.

Baskerville, Richard L. and A. Trevor Wood-Harper (1996). "A critical perspective on action research as a method for information systems research." In: *Journal of Information Technology* 11, pp. 235–246.

— (1998). "Diversity in information systems action research methods." In: *European Journal of Information Systems* 7, pp. 90–107.

Bass, Julian M. (2009). "Empathetic Consultancy: A Reflective Approach to ICTD." In: *Proceedings of the 10th International Conference on Social Implications of Computers in Developing Countries*. Dubai: Dubai School of Government.

Batchelor, S. and P. Norrish (2003). *Sustainable Information and Communication Technology*. URL: `http://www.sustainableicts.org/Sustainable.htm` (visited on July 4, 2011).

Beck, Kent (1999). "Embracing change with extreme programming." In: *IEEE Computer* 32.10, pp. 70–77.

— (2000). *Extreme programming explained: embrace change*. Addison-Wesley Professional.

Beedle, Mike et al. (1999). "SCRUM: An extension pattern language for hyperproductive software development." In: *Pattern Languages of Program Design*. Ed. by Neil Harrison, Brian Foote, and Hans Rohnert. Vol. 4. Addison Wesley, pp. 637–651.

Benbasat, Izak and Robert W. Zmud (1999). "Empirical research in information systems: the practice of relevance." In: *MIS Quarterly* 23.1, pp. 3–16.

— (2003). "The Identity Crisis within the Is Discipline: Defining and Communicating the Discipline's Core Properties." In: *MIS Quarterly* 27.2, pp. 183–194. URL: `http://www.jstor.org/stable/30036527`.

Best, Michael L. (2010). "Understanding Our Knowledge Gaps: Or, Do We Have an ICT4D Field? And Do We Want One?" In: *Information Technologies & International Development* 6.SE, pp. 49–52.

Bieber, Michael et al. (2007). "Towards Systems Design for Supporting Enabling Communities." In: *Journal of Community Informatics* 3.1. URL: `http://ci-journal.net/index.php/ciej/article/viewArticle/281` (visited on Dec. 15, 2011).

BMZ (n.d.). *Information and communication technologies (ICT) – a cross-cutting topic in German development cooperation*. Federal Ministry for Economic Cooperation and Development. URL: http://www.bmz.de/en/what_we_do/issues/wirtschaft/ikt/querschnittsthema/index.html (visited on June 9, 2011).

Boehm, Barry W. (1988). "A spiral model of software development and enhancement." In: *IEEE Computer* 21.5, pp. 61–72.

Bostrom, Robert P. and J. Stephen Heinen (1977). "MIS Problems and Failures: A Socio-Technical Perspective. Part I: The Causes." In: *MIS Quarterly* 1.3, pp. 17–32.

Boyle, Grant (2002). "Putting Context into ICTs in International Development: An Institutional Networking Project in Vietnam." In: *Journal of International Development* 14, pp. 101–112.

Braa, Jørn and Calle Hedberg (2002). "The Struggle for District-Based Health Information Systems in South Africa." In: *The Information Society* 18, pp. 113–127.

Braa, Jørn, Eric Monteiro, and Sundeep Sahay (2004). "Networks of action: sustainable health information systems across developing countries." In: *MIS Quarterly* 28.3, pp. 337–362.

Braa, Jørn et al. (2007). "Scaling up local learning: Experiences from South-South-North Networks of Shared Software Development." In: *Proceedings of the 9th International Conference on Social Implications of Computers in Developing Countries, São Paulo, Brazil*.

Braa, Kristin and Richard Vidgen (1999). "Interpretation, intervention, and reduction in the organizational laboratory: a framework for in-context information system research." In: *Acctounting, Management and Information Technology* 9, pp. 57–47.

Brabander, B. de and G. Thiers (1984). "Successful Information System Development in Relation to Situational Factors Which Affect Effective Communication between MIS-Users and EDP-Specialists." In: *Management Science* 30.2, pp. 137–155.

Brandon, Don (2006). *Project management for modern information systems*. IRM Press. ISBN: 9781591406938. URL: http://books.google.com/books?id=yAUVpzfVAmsC.

Bridges.org (n.d.). *12 Habits of Highly Effective ICT-Enabled Development Initiatives*. URL: http://www.bridges.org/12_habits (visited on July 4, 2011).

Brundtland, Gro Harlem (1987). *Our Common Future: Brundtland Report*. United Nations World Commission on Environment and Development. URL: http://www.

worldinbalance.net/intagreements/1987-brundtland.php (visited on July 4, 2011).

Capiluppi, Andrea et al. (2007). "Adapting the"Staged Model for Software Evolution" to Free/Libre/Open Source Software." In: *IWPSE'07 proceedings*. ACM. Dubrovnik.

Carlsson, Sven A. et al. (Mar. 2011). "Socio-technical IS design science research: developing design theory for IS integration management." In: *Information Systems and E-Business Management* 9.1, pp. 109–131. ISSN: 1617-9846. DOI: 10.1007/s10257-010-0140-6. URL: http://dx.doi.org/10.1007/s10257-010-0140-6.

Carmel, Erran and Ritu Agarwal (2001). "Tactical Approaches for Alleviating Distance in Global Software Development." In: *IEEE Software* 18.2, pp. 22–29.

Cernea, Michael (1993). "The Sociologist's Approach to Sustainable Development." In: *Finance and Development* 30.4, pp. 11–13.

Chege, Mike (2008). "Ubuntuism, commodification, and the software dialectic." In: *First Monday* 13.12. URL: http://frodo.lib.uic.edu/ojsjournals/index.php/fm/article/view/2186/2062 (visited on Dec. 6, 2011).

Chiasson, Mike, Matt Germonprez, and Lars Mathiassen (2009). "Pluralist action research: a review of the information systems literature." In: *Information Systems Journal* 19.1, pp. 31–54. URL: http://dx.doi.org/10.1111/j.1365-2575.2008.00297.x.

Chua, Wai Fong (1986). "Radical Developments in Accounting Thought." In: *The Accounting Review* 61.4, pp. 601–632.

Cisler, Steve (2002). *Schools Online: Planning for Sustainability*. URL: http://geoinfo.uneca.org/sdiafrica/Reference/Ref6/Sustainabilit-booklet.doc (visited on July 4, 2011).

Cockburn, Alistair and Jim Highsmith (2001). "Agile Software Development: The People Factor." In: *IEEE Computer* 34.11, pp. 131–133.

Codd, Edgar F. (1970). "A Relational Model of Data for Large Shared Data Banks." In: *Communications of the ACM* 13.6, pp. 377–387.

— (1982). "Relational Database: A Practical Foundation for Productivity (The 1981 ACM Turing Award Lecture)." In: *Communications of the ACM* 25.2, pp. 109–117.

Cole, Robert et al. (2005). "Being Proactive: Where Action Research meets Design Research." In: *Twenty-Sixth International Conference on Information Systems*. Ed.

by D. Galletta D. Avison and J.I. DeGross. Association for Information Systems. Atlanta, pp. 325–336.

Courant, Paul N. and Rebecca J. Griffiths (2006). *Software and Collaboration in Higher Education: A Study of Open Source Software*. URL: http://www.ithaka.org/ithaka-s-r/strategyold/oss/OOSS_Report_FINAL.pdf (visited on Nov. 11, 2011).

Coward, Chris (2009). *ICT4D, ICTD, or what?* Center for Information & Society. URL: http://chriscoward.wordpress.com/2009/03/11/ict4d-ictd-or-what/ (visited on June 7, 2011).

Curtis, Bill et al. (1987). "On building software process models under the lamppost." In: *ICSE '87 Proceedings of the 9th international conference on Software Engineering*. IEEE Computer Society Press.

Darrow, Ken and Mike Saxenian (n.d.). *Introduction to the Appropriate Technology Sourcebook*. Village Earth. URL: http://villageearth.org/appropriate-technology/appropriate-technology-sourcebook/introduction-to-the-appropriate-technology-sourcebook (visited on Nov. 11, 2011).

Davison, Robert et al. (2000). "Technology Leapfrogging in Developing Countries – An Inevitable Luxury?" In: *Electronic Journal on Information Systems in Developing Countries* 1.5, pp. 1–10.

Davison, Robert M., Maris G. Martinsons, and Ned Kock (2004). "Principles of canonical action research." In: *Information Systems Journal* 14.1, p. 65. URL: http://dx.doi.org/10.1111/j.1365-2575.2004.00162.x.

DeLone, William H. and Ephraim R. McLean (1992). "Information Systems Success: The Quest for the Dependent Variable." In: *Information Systems Research* 3.1, pp. 60–95. URL: http://dx.doi.org/10.1287/isre.3.1.60.

— (2003). "The DeLone and McLean Model of Information Systems Success: A Ten-Year Update." In: *J. of Management Information Systems* 19.4, pp. 9–30. DOI: 10.1.1.88.3031. URL: http://www.jmis-web.org/articles/v19_n4_p9/index.html.

den Hengst, Mariëlle and Gert-Jan de Vreede (2004). "Collaborative Business Engineering: A Decade of Lessons from the Field." In: *Journal of Management Information Systems* 20 (4), pp. 85–114. ISSN: 0742-1222. URL: http://portal.acm.org/citation.cfm?id=1277672.1277677.

Deneulin, Séverine and Lila Shahani (2009). *An Introduction to the Human Development and Capability Approach*. Ottawa, Canada: International Development Research Centre.

Dubé, Line and Guy Paré (2001). *Case Research in Information Systems: Current Practices, Trends, and Recommendations*. Cahier du GReSI no 01-12, ISSN 0832-7203. École des Hautes Études Commerciales de Montréal. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.7705&rep=rep1&type=pdf (visited on Sept. 30, 2011).

Eilhard, Jan (2009). *Open Source Incorporated - Report on Corporate Participation in Open Source Software*. URL: http://ssrn.com/abstract=1360604 (visited on May 30, 2009).

Eisenhardt, Kathleen M. (1989). "Building Theories From Case Study Research." In: *The Academy of Management Review* 14.4, pp. 532–550.

Ezer, Jonathan (2006). "Gandhi's Third Assassination: Information and Communications Technology Education in India." In: *Information Technology for Development* 12.3, pp. 201–212.

Faraj, Samer and Lee Sproull (2000). "Coordinating expertise in software development teams." In: *Management Science* 46.12, pp. 1554–1568.

Favela, Jesús and Feniosky Peña-Mora (2001). "An Experience in Collaborative Software Engineering Education." In: *IEEE Software* 18.2, pp. 47–53.

Fernández, Walter D. (2005). "Information Systems Foundations: Constructing and Criticising." In: ed. by Dennis Hart and Shirley Gregor. Australian National University E-Press. Chap. The grounded theory method and case study data in IS research: issues and design, pp. 43–59.

Fletcher, Peter (1995). "Readers' corner: The role of experiments in computer science." In: *Journal of Systems and Software* 30, pp. 161–163.

Fowler, Martin and Jim Highsmith (2001). *The Agile Manifesto*. Software Development Magazine. URL: http://www.agilemanifesto.org.

Frank, Ulrich (2006). *Towards a pluralistic conception of research methods in information systems research. ICB-Research Report No. 7*. Institut für Informatik und Wirtschaftsinformatik (ICB), Universität Duisburg-Essen. URL: http://users-www.wineme.fb5.uni-siegen.de/home/VolkmarPipek/PUBLIC/workgroup/Papers/Frank2007_PluConcMethISR_Report.pdf (visited on June 19, 2011).

Gallivan, Michael J. and Mark Keil (2003). "The user–developer communication process: a critical case study." In: *Information Systems Journal* 13, pp. 37–68.

Ghosh, Rishab A. (2003). "Licence fees and GDP per capita: The case for open source in developing countries." In: *First Monday* 8.12. URL: http://ojphi.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1103/1023 (visited on Mar. 14, 2012).

— (2004). *The Economics of Free Software and why it matters for developing countries*. USUARIA 2004 – Software Libre Conference. URL: http://flossproject.org/papers/20040527/usuaria-RishabGHOSH-final.pdf (visited on Dec. 6, 2011).

Giddens, A. (1984). *The constitution of society: outline of the theory of structuration*. Cambridge: Polity Press.

Gilb, Tom (1985). "Evolutionary delivery versus the waterfall model." In: *ACM SIGSOFT Software Engineering Notes* 10.3, pp. 49–61.

Glance, David G., Jeremy Kerr, and Alex Reid (2004). "Factors affecting the use of open source software in tertiary education institutions." In: *First Monday* 9.2. URL: http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/viewArticle/1121 (visited on Mar. 5, 2012).

Glaser, Barney G. and Anselm L. Strauss (1967). *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine.

Goldkuhl, Göran (2004). "Design Theories in Information Systems – A Need for Multi-Grounding." In: *Journal of Information Technology Theory and Application* 6.2, pp. 59–72.

Gregg, Dawn G., Uday R. Kulkarni, and Ajay S. Vinzé (2001). "Understanding the Philosophical Underpinnings of Software Engineering Research in Information Systems." In: *Information Systems Frontiers* 3.2, pp. 169–183.

Gregor, Shirley (2006). "The Nature of theory in information systems." In: *MIS Quarterly* 30.3, pp. 611–642.

Gregor, Shirley and David Jones (2007). "The Anatomy of a Design Theory." In: *Journal of the Association for Information Systems* 8.5, pp. 312–335.

Guba, Egon G. and Yvonna S. Lincoln (1994). "Competing paradigms in qualitative research." In: *Handbook of Qualitative Research*. Ed. by N. K. Denzin and Y. S. Lincoln. Sage, pp. 105–117.

Gumucio-Dagron, Alfonso (2003). *What can ICTs do for the rural poor?* Keynote address for the World Summit for the Information Society (WSIS), Geneva, December 11th 2003, at the International Fund for Agricultural Development (IFAD) roundtable: "Six Years Experience in Bridging The Digital Divide". URL: http://www.communicationforsocialchange.org/pdf/what_can_icts_do.pdf (visited on Sept. 29, 2011).

Gurstein, Michael (2003). "Effective use: A community informatics strategy beyond the Digital Divide." In: *First Monday* 8.12. URL: http://firstmonday.org/issues/issue8_12/gurstein/.

— (2007). *What is Community Informatics (and why does it matter)?* Publishing Studies 2. Monza, Italiy: Polimetrica. URL: http://eprints.rclis.org/archive/00012372/01/WHAT_IS_COMMUNITY_INFORMATICS_reading.pdf (visited on June 5, 2009).

Gómez, Ricardo, Juliana Martínez, and Katherine Reilly (2001). "Paths Beyond Connectivity: Experience from Latin America and the Caribbean." In: *Communication South* 1, pp. 110–122.

Hamman, Robin (1997). *Introduction to Virtual Communities Research.* URL: http://www.cybersociology.com/issue_2_virtual_communities/ (visited on Dec. 15, 2011).

Harris, R.W., A. Kumar, and V. Balaji (2003). "Sustainable Telecentres? Two Cases from India." In: *The digital challenge: information technology in the development context.* Ed. by S. Krishna and S. Madon. Ashgate Publishing. Chap. 8, pp. 124–135. URL: http://www.share4dev.info/kb/documents/2237.pdf (visited on July 4, 2011).

Hartwick, Jon and Henri Barki (2001). "Communication as a dimension of user participation." In: *IEEE Transactions on Professional Communication* 44.1, pp. 21–36.

Hecker, Rosalind (1997). "Participatory action research as a strategy for empowering Aboriginal health workers." In: *Australian and New Zealand Journal of Public Health* 21.7, pp. 784–788.

Heeks, Richard (1999). "Software Strategies in Developing Countries." In: *Communications of the ACM* 42.6, pp. 15–20.

— (2001). *"What Did Giddens and Latour Ever Do For Us?": Academic Writings on Information Systems and Development.* Institute for Development Policy and

Management, University of Manchester. URL: http://www.sed.manchester.ac.uk/idpm/research/publications/wp/di/short/di_sp06.pdf (visited on June 17, 2011).

Heeks, Richard (2002a). "i-development not e-development: special issue on ICTs and development." In: *Journal of International Development* 14.1, pp. 1–11.

— (2002b). "Information Systems and Developing Countries: Failure, Success, and Local Improvisations." In: *The Information Society* 18, pp. 101–112. DOI: 10.1080/01972240290075039.

— (2005a). *Free and Open Source Software: A Blind Alley for Developing Countries?* Institute for Development Policy and Management, University of Manchester. URL: http://www.sed.manchester.ac.uk/idpm/research/publications/wp/di/short/DIGBriefing1FOSS.pdf (visited on Dec. 6, 2011).

— (2005b). *Sustainability and the future of e-development. eDevelopment Briefing No. 10*. Institute for Development Policy and Management, University of Manchester. URL: http://www.sed.manchester.ac.uk/idpm/research/publications/wp/di/short/DIGBriefing10Sustain.pdf (visited on July 4, 2011).

— (2006). "Theorizing ICT4D Research." In: *Information Technologies and International Development* 3.3, pp. 1–4.

— (2008). "ICT4D 2.0: The Next Phase of Applying ICT for International Development." In: *IEEE Computer* 41.6, pp. 26–33. URL: http://dx.doi.org/10.1109/MC.2008.192.

— (2010). "Do information and communication technologies (ICTs) contribute to development?" In: *J. Int. Dev.* 22, pp. 625–640. DOI: 10.1002/jid.1716.

Herbsleb, James D. and Deependra Moitra (2001). "Global Software Development." In: *IEEE Software* 18.2, pp. 16–20.

Hevner, Alan R. and Salvatore T. March (2003). "The Information Systems Research Cycle." In: *IEEE Computer* 36.11, pp. 111–113. DOI: 10.1109/MC.2003.1244541.

Hevner, Alan R. et al. (2004). "Design Science in Information Systems Research." In: *MIS Quarterly* 28.1, pp. 75–105. URL: http://dblp.uni-trier.de/rec/bibtex/journals/misq/HevnerMPR04.

Hirschheim, Rudy and Heinz K. Klein (1989). "Four Paradigms of Information Systems Development." In: *Communications of the ACM* 32.10, pp. 1199–1216.

Hofstede, Geert H. (1980). *Cultural Consequences: International Differences in Work Related Values*. Sage.

— (1991). *Cultures and Organizations: Software of the Mind*. McGraw-Hill.

Hollick, Malcolm (1982). "The Appropriate Technology Movement and Its Literature: A Retrospective." In: *Technology in Society* 4, pp. 213–229.

Holmström, Helena et al. (2006). "Agile Practices Reduce Distance in Global Software Development." In: *Information Systems Management* 23.3, pp. 7–18.

IISD (n.d.). *What is Sustainable Development?* International Institute for Sustainable Development (IISD). URL: `http://www.iisd.org/sd/` (visited on July 4, 2011).

Iivari, Juhani (2007). "A Paradigmatic Analysis of Information Systems As a Design Science." In: *Scandinavian Journal of Information Systems* 19.2, pp. 39–64.

Iivari, Juhani, Rudy Hirschheim, and Heinz K. Klein (2004). "Towards a distinctive body of knowledge for Information Systems experts: coding ISD process knowledge in two IS journals." In: *Information Systems Journal* 14, pp. 313–342.

Iivari, Juhani and John Venable (2009). "Action Research and Design Science Research – Seemingly similar but decisively dissimilar." In: *17th European Conference on Information Systems*. Verona.

Informatics defined. Indiana University School of Informatics. URL: `http://informatics.iupui.edu/about/what-is-informatics/` (visited on Mar. 5, 2012).

Johanson, Graeme (2010). "Delineating the Meaning and Value of Development Informatics." In: *ICTs and Sustainable Solutions for the Digital Divide: Theory and Perspectives*. Ed. by Jacques Steyn and Graeme Johanson. IGI Global. Chap. 1.

Jones, Matthew R. and Helena Karsten (2008). "Giddens's Structuration Theory and Information Systems Research." In: *MIS Quarterly* 32.1, pp. 127–157. URL: `http://misq.org/gidden-s-structuration-theory-and-information-systems-research.html`.

Joubert, Pieter (2007). "Minimum Critical Technical Success Factors for e-development projects." In: *Proceedings of the 9th International Conference on Social Implications of Computers in Developing Countries*. São Paulo, Brazil.

Järvinen, Pertti (2007). "Action Research is Similar to Design Science." In: *Quality and Quantity* 41, pp. 37–54.

Kahen, Goel (1995). "Assessment of Information Technology for Developing Countries: Appropriateness, Local Constraints, IT Characteristics and Impacts." In: *International Journal of Computer and Applications Technology* 8.5/6, pp. 325–332.

Kala, Rahul (2008). *The Open and Closed Styles of Talent Acquisition and Management: Which takes an edge where*. Students' Forum for Free/Open Source Software. Indian Institute of Information Technology and Management. URL: http://ifipwg213.org/system/files/paper.pdf (visited on Mar. 14, 2012).

Keats, Derek (2007). *A network for Capacity-Building in Software Engineering through Free Software development in Africa*. URL: http://www.slideshare.net/dkeats/a-network-for-capacitybuilding-in-software-engineering-in-africa (visited on Sept. 27, 2011).

Kimaro, Honest C. and José L. Nhampossa (2004). "The challenges of sustainability of health information systems in developing countries: comparative case studies of Mozambique and Tanzania." In: *ECIS 2004 Proceedings*.

Kimaro, Honest C. and José Leopoldo Nhampossa (2005). "Analyzing the Problem of Unsustainable Health Information Systems in Less-Developed Economies: Case Studies From Tanzania and Mozambique." In: *Information Technology for Development* 11.3, pp. 273–198. DOI: 10.1002/itdj.20016.

Klein, Heinz K. and Michael D. Myers (1999). "A set of principles for conducting and evaluating interpretive field studies in information systems." In: *MIS Quarterly* 23 (1), pp. 67–93. ISSN: 0276-7783. DOI: 10.2307/249410. URL: http://portal.acm.org/citation.cfm?id=311375.311384.

Kling, Rob and Roberta Lamb (1999). "IT and Organizational Change in Digital Economies: A Socio-Technical Approach." In: *Computers and Society* 29.3, pp. 17–25.

Kock, Ned et al. (2002). "IS Research Relevance Revisited: Subtle Accomplishment, Unfulfilled Promise, or Serial Hypocrisy?" In: *Communications of the Association for Information Systems* 8, pp. 330–346.

Kogut, Bruce and Anca Metiu (2001). "Open-source Software Development and Distributed Innovation." In: *Oxford Review of Economic Policy* 17.2, pp. 248–264.

Korpela, Mikko, Anja Mursu, and H.A. Soriyan (2002). "Information Systems Development as an Activity." In: *Computer Supported Cooperative Work* 11.1–2 (1). 10.1023/A:1015252806306, pp. 111–128. ISSN: 0925-9724. URL: http://dx.doi.org/10.1023/A:1015252806306.

Kraut, Robert E. and Lynn A. Streeter (1995). "Coordination in software development." In: *Communications of the ACM* 38.3, pp. 69–81.

Krishna, S. and Geoff Walsham (2005). "Implementing public information systems in developing countries: learning from a success story." In: *Information Technology for Development* 11 (2), pp. 123–140. ISSN: 0268-1102. DOI: 10.1002/itdj.20007. URL: http://portal.acm.org/citation.cfm?id=1146091.1146094.

Krishnamurthy, Sandeep (2003). "An Analysis of Open Source Business Models." In: *Perspectives on Free and Open Source Software*. Ed. by Joseph Feller et al. MIT Press. Chap. 15, pp. 279–296. URL: http://ssrn.com/abstract=650001 (visited on May 24, 2009).

Kuechler, Bill and Vijay Vaishnavi (2008a). "On theory development in design science research: anatomy of a research project." In: *European Journal of Information Systems* 17, pp. 489–504.

Kuechler, William and Vijay Vaishnavi (2008b). "The emergence of design research in information systems in North America." In: *Journal of Design Research* 7.1, pp. 1–16.

Kumar, Richa (2004). "eChoupals: A Study on the Financial Sustainability of Village Internet Centers in Rural Madhya Pradesh." In: *Information Technologies and International Development* 2.1, pp. 45–73.

Lange, Carola (2006). *Entwicklung und Stand der Disziplinen Wirtschaftsinformatik und Information Systems*. Institut für Informatik und Wirtschaftsinformatik (ICB), Universität Duisburg-Essen. URL: http://www.wi-inf.uni-duisburg-essen.de/FGFrank/documents/Arbeitsberichte_ICB/No4.pdf (visited on June 19, 2011).

Larman, Craig and Victor R. Basili (2003). "Iterative and Incremental Development: A Brief History." In: *IEEE Computer* 36.6, pp. 47–56.

Latour, Bruno (1996). *On actor-network theory*. URL: http://www9.georgetown.edu/faculty/irvinem/theory/Latour-clarifications.pdf (visited on Mar. 7, 2012).

Layman, Lucas et al. (2006). "Essential communication practices for Extreme Programming in a global software development team." In: *Information and Software Technology* 48, pp. 781–794.

Lee, Allen S. (2000). *Systems Thinking, Design Science, and Paradigms: Heeding Three Lessons from the Past to Resolve Three Dilemmas in the Present to Direct a Trajectory for Future Research in the Information Systems Field*. URL: http://www.people.vcu.edu/~aslee/ICIM-keynote-2000/ICIM-keynote-2000.htm (visited on Feb. 4, 2012).

— (2001). "Editorial." In: *MIS Quarterly* 25.1, pp. iii–vii.

Lehman, Meir M. (1996). "Feedback in the software evolution process." In: *Information and Software Technology* 38, pp. 681–686.

Lehman, Meir M. and Juan F. Ramil (2001). "Rules and Tools for Software Evolution Planning and Management." In: *Annals of Software Engineering* 11, pp. 15–44.

Lerner, Josh and Jean Tirole (2002). "Some Simple Economics of Open Source." In: *The Journal of Industrial Economics* 50.2, pp. 197–234.

— (2005). "The Scope of Open Source Licensing." In: *Journal of Law Economics & Organization* 21.1, pp. 20–56.

Lin, Nan (1999). "Building a Network Theory of Social Capital." In: *Connections* 22.1, pp. 28–51.

Lindberg, Van (2008). *Intellectual Property and Open Source - A Practical Guide to Protecting Code*. O'Reilly.

Liu, Wei and Chris Westrup (2003). "ICTs and Organizational Control across Cultures." In: *Organizational information systems in the context of globalization: WG8.2 & WG9.4 Working Conference on Information Systems Perspectives and Challenges in the Context of Globalization, June 15-17, 2003, Athens, Greece*. Ed. by Angeliki Poulymenakou Mikko Korpela Ramiro Montealegre. URL: http://books.google.com/books?id=I9TpR5A188oC&lpg=PA155&ots=HAqURNK5CE&dq=liu%20westrup%20ICTs%20and%20organizational%20control%20across%20cultures&lr&pg=PA155#v=onepage&q&f=false (visited on Sept. 30, 2011).

Loukola, Olli and Simo Kyllönen (2005). "Sustainable use of renewable natural resources – from principles to practices." In: ed. by A. Jalkanen and P. Nygren. Helsinki, Finland: Department of Forest Ecology, University of Helsinki. Chap. The philosophies of sustainability, pp. 1–7.

Luna-Reyes, Luis F. et al. (2005). "Information systems development as emergent socio-technical change: a practice approach." In: *European Journal of Information Systems* 14, pp. 93–105.

MacCormack, Alan, Roberto Verganti, and Marco Jansiti (2001). "Developing Products on "Internet Time": The Anatomy of a Flexible Development Process." In: *Management Science* 47.1, pp. 133–150.

Macome, Esselina (2008). "On Implementation of an Information System in the Mozambican Context: The EDM Case Viewed Through ANT Lenses." In: *Information Technology for Development* 14.2, pp. 154–170. DOI: 10.1002/itdj.20063.

Madon, Shirin (2005). "The Internet and socioeconomic development: exploring the interaction." In: *Information Technology & People* 13.2, pp. 85–101.

March, Salvatore T. and Gerald F. Smith (1995). "Design and natural science research on information technology." In: *Decision Support Systems* 15, pp. 251–266.

Markus, M. Lynne (2007). "The governance of free/open source software projects: monolithic, multidimensional, or configurational?" In: *Journal of Management and Governance* 11, pp. 151–163.

Markus, M. Lynne and Robert I. Benjamin (1997). "The Magic Bullet Theory in IT-Enabled Transformation." In: *Sloan Management Review* 38.2, pp. 55–68. URL: http://dialnet.unirioja.es/servlet/articulo?codigo=2511876.

Markus, M. Lynne, Ann Majchrzak, and Les Gasser (2002). "A Design Theory for Systems That Support Emergent Knowledge Processes." In: *MIS Quarterly* 26.3, pp. 179–212.

Markus, M. Lynne and Daniel Robey (1988). "Information Technology and Organizational Change: Causal Structure in Theory and Research." In: *Management Science* 34.5, pp. 583–598.

Martin, Patricia Yancey and Barry A. Turner (1986). "Grounded Theory and Organizational Research." In: *Journal of Applied Behavioral Science* 22, pp. 141–157.

Massingue, Venâncio Simão (2003). "Building Awareness and Supporting African Universities in ICT Management: The Big ICT Five (Strategy, Development/Acquisition, Implementation, Utilization, Service Management)." PhD thesis. Delft University of Technology.

McDonald, Robert H. (2009). "Sustainable Communities for Software: Variations on a Theme." In: *Educause Review* 44.5, pp. 6–7.

McIver, William Jr. (2003). *A community informatics for the information society.* UNRISD Briefing Paper. UNRISD. URL: http://comunica.org/com_rights/mciver.pdf (visited on June 27, 2009).

McKay, Judy and Peter Marshall (2001). "The dual imperatives of action research." In: *IT & People* 14.1, pp. 46–59. URL: http://dx.doi.org/10.1108/09593840110384771.

— (2005). "A Review of Design Science in Information Systems." In: *ACIS 2005 Proceedings*. URL: http://aisel.aisnet.org/acis2005/5.

— (2007). "Science, Design, and Design Science: Seeking Clarity to Move Design Science Research Forward in Information Systems." In: *ACIS 2007 Proceedings*. URL: http://aisel.aisnet.org/acis2007/55.

McRobie, George (1979). "Intermediate Technology: Small Is Successful." In: *Third World Quarterly* 1.2, pp. 71–86.

Mills, Harlan D. (1976). "Software Development." In: *IEEE Transactions on Software Engineering* SE-2, p. 4.

Ministério de Educação e Cultura (2005). *Estatísticas e Indicadores do Ensino Superior 2004*. URL: http://www.mec.gov.mz/documento.php?id=447 (visited on July 27, 2009).

— (2006). *Estudantes por Área Científica 2005 e 2006*. URL: http://www.mec.gov.mz/documento.php?id=445 (visited on July 27, 2009).

Moens, Nicolaas Paul (2010). "Innovating in sectoral governance and development with ICT in agriculture, education and health: The Round Table Process." PhD thesis. Vrije Universiteit.

Moens, Nicolaas Paul et al. (2010). "A Constructive Technology Assessment Approach to ICT Planning in Developing Countries: Evaluating the First Phase, the Roundtable Workshop." In: *Information Technology for Development* 16.1, pp. 34–61.

Montealegre, Ramiro (1999). "A case for more case study research in the implementation of information technology in less-developed countries4." In: *Information Technology for Development* 8 (4), pp. 199–207. ISSN: 0268-1102. DOI: http://dx.doi.org/10.1080/02681102.1999.9525310. URL: http://dx.doi.org/10.1080/02681102.1999.9525310.

Morawczynski, Olga and Ojelanki Ngwenyama (2007). "Unraveling the Impact of Investments in ICT, Education and Health on Development: An Analysis of Archival Data of Five West African Countries Using Regression Splines." In: *The Electronic Journal of Information Systems in Developing Countries* 29.5, pp. 1–15. URL: http://www.ejisdc.org/ojs2/index.php/ejisdc/article/view/352.

Morrison, Joline and Joey F. George (1995). "Exploring the Software Engineering Component in MIS Research." In: *Communications of the ACM* 38.7, pp. 80–91.

Mosse, Emilio and Sundeep Sahay (2003). "Counter Networks, Communication and Health Information Systems: A Case Study from Mozambique." In: *Information Systems Perspectives and Challenges in the Context of Globalization*. Ed. by Mikko Korpela, Ramiro Montealegre, and Angeliki Poulymenakou. Vol. 254. IFIP Conference Proceedings. Kluwer, pp. 35–51. ISBN: 1-4020-7488-3.

Mulira, Nora Kasirye (2007). "Implementing inter-organisational service systems: An approach for emerging networks in volatile contexts." PhD thesis. Technische Universiteit Delft.

Myers, Michael D. (1997). "Qualitative Research in Information Systems." In: *MIS Quarterly* 21.2, pp. 241–242. URL: `http://www.qual.auckland.ac.nz/` (visited on Aug. 20, 2011).

Myers, Michael D. and Felix B. Tan (2003). "Beyond Models of National Culture in Information Systems Research." In: *Advanced topics in global information management*. Ed. by Felix B. Tan. IGI Publishing, pp. 14–29.

NACI (2002). *Open Software & Open Standards in South Africa*. National Advisory Council on Innovation, Open Software Working Group. URL: `http://www.naci.org.za/` (visited on May 26, 2010).

Nauman, Abou Bakar, Romana Aziz, and A. F. M . Ishaq (2005). "Information Systems Development Failure: A Case Study to Highlight the IS Development Complexities in Simple, Low Risk Projects in Developing Countries." In: *The Second International Conference on Innovations in Information Technology*.

Ngwenyama, Ojelanki et al. (2006). "Is there a relationship between ICT, health, education and development? An empirical analysis of five west african countries from 1997–2003." In: *The Electronic Journal of Information Systems in Developing Countries* 23.5, pp. 1–11. URL: `http://www.ejisdc.org/ojs2/index.php/ejisdc/article/view/189`.

Nhampossa, José Leopoldo (2005). "Re-thinking technology transfer as technology translation: A case study of health information systems in Mozambique." PhD thesis. University of Oslo.

Nicholson, Brian and Sundeep Sahay (2001). "Some Political and Cultural Issues in the Globalisation of Software Development: Case experience from Britain and India." In: *Information and Organization* 11, pp. 25–43.

Niehaves, Björn (2007). "On Epistemological Pluralism in Design Science." In: *Scandinavian Journal of Information Systems* 19.2, pp. 99–110. URL: http://aisel.aisnet.org/sjis/vol19/iss2/7.

Nuscheler, Franz (2004). *Entwicklungspolitik*. 5th ed. Bonn: Verlag J.H.W. Dietz.

Nygaard, Kristen (2002). "Foreword." In: *The Labyrinths of Information*. Ed. by Claudio Ciborra. Oxford University Press, pp. v–x.

Okunoye, Adekunle and Helena Karsten (2003). "Global access to knowledge: Findings from academic research organisations in sub-Saharian Africa." In: *Information Technology & People* 16.3, pp. 353–373.

Orlikowski, Wanda J. (1992). "The Duality of Technology: Rethinking the Concept of Technology in Organizations." In: *Organization Science* 3.3, pp. 398–427. ISSN: 10477039. DOI: 10.2307/2635280. URL: http://dx.doi.org/10.2307/2635280.

— (2000). "Using Technology and Constituting Structures: A Practice Lens for Studying Technology in Organizations." In: *Organization Science* 11.4, pp. 404–428. DOI: 10.2307/2640412. URL: http://dx.doi.org/10.2307/2640412.

Orlikowski, Wanda J. and Stephen R. Barley (2001). "Technology and Institutions: What Can Research on Information Technology and Research on Organizations Learn from Each Other?" In: *MIS Quarterly* 25.2, pp. 145–165.

Orlikowski, Wanda J. and Jack J. Baroudi (1991). "Studying Information Technology in Organizations: Research Approaches and Assumptions." In: *Information Systems Research* 2.1, pp. 1–28. URL: http://dx.doi.org/10.1287/isre.2.1.1.

Orlikowski, Wanda J. and J. Debra Hofman (1997). "An Improvisational Model for Change Management: The Case of Groupware Technologies." In: *Sloan Management Review* 38.2, pp. 11–21.

Orlikowski, Wanda J. and C. Suzanne Iacono (2001). "Research commentary: Desperately seeking 'IT' in IT research - A call to theorizing the IT Artifact." In: *Information Systems Research* 12.2, pp. 121–134.

Orlikowski, Wanda J. and Daniel Robey (1991). "Information Technology and the Structuring of Organizations." In: *Information Systems Research* 2.2, pp. 143–169. URL: http://dx.doi.org/10.1287/isre.2.2.143.

Parmar, Vikram (2009). "A Multidisciplinary Approach to ICT Development." In: *Information Technologies and International Development* 5.4, pp. 89–96.

Parnas, David Lorge (1972). "On the Criteria to be used in Decomposing Systems into Modules." In: *Communications of the ACM* 15.12, pp. 1053–1058.

Parnas, David Lorge and Paul C. Clements (1986). "A rational design process: How and why to fake it." In: *IEEE Transactions on Software Engineering* 12.2, pp. 251–257.

Pfeffers, Ken et al. (2007). "A Design Science Research Methodology for Information Systems Research." In: *Journal of Management Information Systems* 24.3, pp. 45–77. DOI: 10.2753/MIS0742-1222240302.

Pluye, Pierre, Louise Potvin, and Jean-Louis Denis (2004). "Making public health programs last: conceptualizing sustainability." In: *Evaluation and Program Planning* 27, pp. 121–133.

Pluye, Pierre et al. (2004). "Program sustainability: focus on organizational routines." In: *Health Promotion International* 19.4, pp. 489–500. DOI: 10.1093/heapro/dah411.

Polak, Paul (Sept. 2010). *The Death of Appropriate Technology I: If you can't sell it don't do it.* URL: http://blog.paulpolak.com/?p=376 (visited on Aug. 30, 2011).

Portes, Alejandro (2000). "The Two Meanings of Social Capital." In: *Sociological Forum* 15.1, pp. 1–12.

Prasad, Acklesh (2009). "Understanding successful use of technology in organisations in developing countries: A structurational perspective." In: *Electronic Journal on Information Systems in Developing Countries* 37.3, pp. 1–9.

Pscheidt, Markus (2008). "Sustainability Factors for Information Systems in Developing Countries – Academic Registry Information System in Mozambique." In: *Prato Community CIRN Conference 2008: ICTs for Social Inclusion: What is the Reality?* Prato, Italy.

— (2011). "Structurational analysis of cross-cultural development of an academic registry information system in Mozambique." In: *Information Technology for Development* 17.3, pp. 168–186.

Pscheidt, Markus, Eduard J. Simons, and Theo van der Weide (2009). "Towards 'best practices' in North-South Open Source projects – lessons learned from the ARIS project in Mozambique." In: *3rd IDIA Conference: Digitally Empowering Communities: Learning from Development Informatics Practice.* Berg-en-dal, Kruger, South Africa.

Pscheidt, Markus and Theo van der Weide (2009). "Supporting the ARIS community system in Mozambique." In: *Prato Community CIRN Conference 2009: Empowering communities: learning from community informatics practice*. Prato, Italy.

— (2010a). "Bridging the Digital Divide by Open Source." In: *International Journal of Innovation in the Digital Economy, Special Issue on Digital Divide* 1.2, pp. 44–60.

— (2010b). "Sustainability of collaborative information system development projects – a North-South case study." In: *4th IDIA Conference: Exploring Success and Failure in Development Informatics: Innovation, Research and Practice*. Cape Town, South Africa.

Pscheidt, Markus, Victor van Reijswoud, and Theo van der Weide (2009). "Assessing Appropriate ICT with ARIS case in Mozambique." In: *ICCIR'09: 5th Annual International Conference on Computing and ICT Research*. Makerere University. Kampala, Uganda.

Rajani, Niranjan (2003). *Free as in Education: Significance of the Free/Libre and Open Source Software for Developing Countries*. Helsinki, Finland. URL: `http://www.tuxcafe.org/~renee/universitefem06/free_as_in_education_niranjan.pdf` (visited on Mar. 14, 2012).

Rajlich, Václav T. and Keith H. Bennett (2000). "A Staged Model for the Software Life Cycle." In: *IEEE Computer* 33.7, pp. 66–71.

Rapoport, Robert N. (1970). "Three Dilemmas in Action Research: With Special Reference to the Tavistock Experience." In: *Human Relations* 23.6, pp. 499–513. DOI: `10.1177/001872677002300601`.

Raymond, Eric Steven (1998). "Homesteading the Noosphere." In: *First Monday* 3.10. URL: `http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/viewArticle/621/542` (visited on Dec. 3, 2012).

— (2005). "The Cathedral and the Bazaar." In: *First Monday*.

Riehle, Dirk (2007). "The Economic Motivation of Open Source Software: Stakeholder Perspectives." In: *IEEE Computer* 40.4, pp. 25–32.

Rogers, Everett M. (2002). "The Nature of Technology Transfer." In: *Science Communication* 23, pp. 323–341.

— (2003). *Diffusion of Innovations*. 5th. New York: The Free Press. ISBN: 0743222091.

Rolland, Knut H. and Eric Monteiro (2002). "Balancing the Local and the Global in Infrastructural Information Systems." In: *The Information Society* 18, pp. 87–100.

Rose, Jeremy and Rens Scheepers (2001). "Structuration Theory and Information System Development - Frameworks for Practice." In: *ECIS*. URL: http://is2.lse.ac.uk/asp/aspecis/20010096.pdf.

Rosemann, Michael and Iris Vessey (2008). "Toward improving the relevance of information systems research to practice: the role of applicability checks." In: *MIS Quarterly* 32.1, pp. 1–22.

Royce, Winston W. (1970). "Managing the development of large software systems." In: *Proceedings IEEE WESTCON*, pp. 328–338.

Sahay, Sundeep and Chrisanthi Avgerou (2002). "Introducing the special issue on information and communication technologies in developing countries." In: *The Information Society* 18, pp. 73–76. DOI: 10.1080/01972240290075002.

Sangmeister, Hartmut (1998). "Book review: Reinhard Stockmann: Die Wirksamkeit der Entwicklungshilfe. Eine Evaluation der Nachhaltigkeit von Programmen und Projekten der Berufsbildung." In: *Politik und Gesellschaft* 3. URL: http://www.fes.de/ipg/ipg3_98/rezsangmeister.html (visited on Oct. 1, 2011).

Saraswati, Baidyanath (1997). "Cultures and Development: Guideline Questions." In: *Integration of Endogenous Cultural Dimension Into Development*. Ed. by Baidyanath Saraswati. New Delhi: D. K. Printworld Pvt. Ltd. Chap. 2. URL: http://ignca.nic.in/cd_05005.htm (visited on Apr. 10, 2011).

Sayed, Heba El and Chris Westrup (2003). "Egypt and ICTs: How ICTs bring national initiatives, global organizations and local companies together." In: *Information Technology & People* 16.1, pp. 76–92.

Schuftan, Claudio (1996). "The community development dilemma: What is really empowering?" In: *Community Development Journal* 31.3, pp. 260–264.

Schumacher, Ernst F. (1973). *Small is Beautiful*. URL: http://www.rmutphysics.com/charud/oldnews/227/small.pdf (visited on July 28, 2009).

Schunter, Johannes (2007). "ICT4D – Entwicklungsarbeit ohne Entwicklungstheorie? Theoretische und strategische Rahmenkonzeptionen für die Rolle von Informations- und Kommunikationstechnologien in der Entwicklungszusammenarbeit." MA thesis. Universität Stuttgart, Institut für Sozialwissenschaften, Abteilung für Internationale Beziehungen und Europäische Integration.

Seddon, Peter B. et al. (1999). "Dimensions of information systems success." In: *Communications of AIS* 2, pp. 1–61. URL: http://portal.acm.org/citation.cfm?id=374477\&coll=portal\&dl=ACM.

Sen, Amartya Kumar (1999). *Development as freedom*. Oxford University Press, p. 366.

Shah, Sonali K. (2006). "Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development." In: *Management Science* 52.7, pp. 1000–1014.

Silva, Leiser and Eugenio Figueroa (2002). "Institutional intervention and the expansion of ICTs in Latin America: The case of Chile." In: *Information Technology & People* 15.1, pp. 8–25.

Simon, Herbert Alexander (1996). *The sciences of the artificial*. 3rd ed. Massachusetts Institute of Technology.

Simpson, Lyn (2005). "Community Informatics and Sustainability: Why Social Capital Matters." In: *The Journal of Community Informatics* 1.2, pp. 102–119.

Smith, Mark K. (2001). *Kurt Lewin: Groups, Experiential Learning and Action Research*. URL: http://www.infed.org/thinkers/et-lewin.htm (visited on Apr. 4, 2011).

Stallman, Richard (1996). *What is free software?* Free Software Foundation. URL: http://www.gnu.org/philosophy/free-sw.html (visited on Dec. 2, 2011).

— (2002). "Selling Free Software." In: *Free Software, Free Society: Selected Essays of Richard M. Stallman*. Ed. by Joshua Gay. GNU Press. Chap. 8, pp. 65–67. URL: http://www.gnu.org/philosophy/selling.html (visited on Dec. 6, 2011).

— (2007). *Why Open Source misses the point of Free Software*. Free Software Foundation. URL: http://www.gnu.org/philosophy/open-source-misses-the-point.html (visited on Dec. 2, 2011).

Stillman, Larry Jeffrey Hirsh (2006). "Understandings of Technology in Community-Based Organisations: A Structurational Analysis." PhD thesis. Monash University.

Stockmann, Reinhard (1996). *Die Wirksamkeit der Entwicklungshilfe. Eine Evaluation der Nachhaltigkeit von Programmen und Projekten der Berufsbildung*. Westdeutscher Verlag, p. 484.

— (2000). "Wirkungsevaluation in der Entwicklungspolitik." In: *Vierteljahrshefte zur Wirtschaftsforschung* 69.3, pp. 438–452.

Stoecker, Randy (2005). "Is Community Informatics good for communities? Questions confronting an emerging field." In: *Journal of Community Informatics* 1.3. URL: http:

//www.ci-journal.net/index.php/ciej/article/viewArticle/183 (visited on Dec. 15, 2011).

Susman, Gerald I. and Roger D. Evered (1978). "An Assessment of the Scientific Merits of Action Research." In: *Administrative Science Quarterly* 23.4, pp. 582–603. URL: http://www.jstor.org/stable/2392581.

Sutinen, Erkki and Matti Tedre (2010). "ICT4D: A Computer Science Perspective." In: *Algorithms and Applications: Essays Dedicated to Esko Ukkonen on the Occasion of His 60th Birthday*. Ed. by Tapio Elomaa, Heikki Mannila, and Pekka Orponen. Springer-Verlag Berlin Heidelberg, pp. 221–231.

Swartout, William and Robert Balzer (1982). "On the Inevitable Intertwining of Specification and Implementation." In: *Communications of the ACM* 25.7, pp. 438–440.

Takeuchi, Hirotaka and Ikujiro Nonaka (1986). "The new new product development game." In: *Harvard Business Review* 3.3, pp. 137–146.

Tate, Kevin (2006). *Sustainable Software Development: An Agile Perspective*. Addison-Wesley Professional.

Tedre, Matti (2009). "The Social Study of Computer Science." In: *Handbook of research on socio-technical design and social networking systems*. Ed. by Brian Whitworth and Aldo de Moor. Vol. 1. IGI Global. Chap. 2, pp. 23–38.

Tharakan, John (2006). "Educating Engineers in Appropriate Technology for Development." In: *9th UICEE Annual Conference on Engineering Education*. Muscat, Oman.

Thompson, Mark (2004). "Discourse, 'Development' & the 'Digital Divide': ICT & the World Bank." In: *Review of African Political Economy* 31.99, pp. 103–123.

Thompson, Mark P.A. (2002). "Cultivating meaning: interpretive fine-tuning of a South African health information system." In: *Information and Organization* 12, pp. 183–211.

Tong, Tan Wooi (2004). *Free/Open Source Software Education*. United Nations Development Programme-Asia Pacific Development Information Programme.

Toyama, Kentaro and M. Bernardine Dias (2008). "Guest Editors' Introduction: Information and Communication Technologies for Development." In: *Computer*, pp. 22–25.

UNECA (2010). *Free and Open Source Software: an African Regional perspective*. United Nations Economic Commission for Africa. URL: http://www.uneca.org/eca_programmes/it_for_development/events/accra/AfricanLanguages/CEA_FOSS_African_Regional_perspectives.ppt (visited on May 26, 2010).

Unwin, Tim (2009). *ICT4D: Information and Communication Technology for Development*. 1st ed. Cambridge University Press. ISBN: 052171236X. URL: http://www.worldcat.org/isbn/052171236X.

Uwadia, Charles O. et al. (2006). "Risk Factors in the Collaborative Development of Information Systems for Nigerian Universities." In: *Information Technology for Development* 12.2, pp. 91–111.

van Aken, Joan E. (2004). "Management Research Based on the Paradigm of the Design Sciences: The Quest for Field-Tested and Grounded Technological Rules." In: *Journal of Management Studies* 41.2, pp. 219–246.

van Reijswoud, Victor (2009). "Appropriate ICT as a Tool to Increase Effectiveness in ICT4D: Theoretical considerations and illustrating cases." In: *The Electronic Journal of Information Systems in Developing Countries* 38, pp. 1–18.

van Reijswoud, Victor and Arjan de Jager (2008). *Free and Open Source Software for development: exploring expectations, achievements and the future*. Ed. by Giandomenico Sica. Publishing Studies 5. Monza, Italiy: Polimetrica.

van Reijswoud, Victor and Emmanuel Mulo (2006). "Applying Open Source Software in a Development Context: expectations and experiences. A Case Study of a University in Uganda." In: *E-Learning* 3.3, pp. 361–372.

van Vliet, Hans (2008). *Software Engineering: Principles and Practice*. 3rd. Wiley Publishing. ISBN: 0470031468.

Vaughan, Donna (2006). *ICT4D – Linking Policy to Community Outcomes*. Partners in Micro-development Inc. URL: http://www.microdevpartners.org/documents/ICT4DLinkingPolicytoCommunityOutcomesPDF.pdf (visited on June 26, 2009).

Venable, John R. (2006). "The Role of Theory and Theorising in Design Science Research." In: *Proceedings of the First International Conference on Design Science Research in Information Systems and Technology (DESRIST)*. Claremont, CA, pp. 1–18.

Villanueva, Edgar David (2002). *Open letter to Microsoft Peru*. URL: http://www.theregister.co.uk/content/4/25157.html (visited on Dec. 6, 2011).

Virnoche, Mary E. and Gary T. Marx (1997). ""Only Connect"-E. M. Forster in an Age of Electronic Communication: Computer-Mediated Association and Community Networks." In: *Sociological Inquiry* 67.1, pp. 85–100.

von Hippel, Eric (2001). "Innovation by User Communities: Learning from Open-Source Software." In: *MIT Sloan Management Review* 42, pp. 82–86.

— (2003). “Open Source Software and the “Private-Collective” Innovation Model: Issues for Organization Science.” In: *Organization Science* 14.2, pp. 209–223.

— (2007). “Horizontal innovation networks—by and for users.” In: *Industrial and Corporate Change* 16.2, pp. 293–315.

von Hippel, Eric, Jeroen P.J. de Jong, and Stephen Flowers (2011). *Comparing Business and Household Sector Innovation in Consumer Products: Findings from a Representative Study in the UK*. URL: http://ssrn.com/abstract=1683503 (visited on Dec. 10, 2011).

Wade, Robert Hunter (2002). “Bridging the digital divide: New route to development or new form of dependency.” In: *Global governance* 8, pp. 443–466.

Walls, Joseph G., George R. Widmeyer, and Omar A. El Sawy (1992). “Building an Information System Design Theory for Vigilant EIS.” In: *Information Systems Research* 3.1, pp. 36–59.

Walsham, Geoff (1993). *Interpreting Information Systems in Organizations*. 1st. New York, NY, USA: John Wiley & Sons, Inc. ISBN: 0471938149.

— (1995a). “Emergence of interpretivism in IS research.” In: *Information Systems Research* 6.4, pp. 376–394.

— (1995b). “Interpretive case studies in IS research: nature and method.” In: *European Journal of Information Systems* 4, pp. 74–81. URL: http://dx.doi.org/10.1057/ejis.1995.9.

— (2002). “Cross-Cultural Software Production and Use: A Structurational Analysis.” In: *MIS Quarterly* 26.4, pp. 359–380. ISSN: 02767783. DOI: 10.2307/4132313. URL: http://dx.doi.org/10.2307/4132313.

— (2005). “Development, global futures and IS research: a polemic.” In: *Journal of Strategic Information Systems* 14, pp. 5–15.

— (2006). “Doing interpretive research.” In: *European Journal of Information Systems* 15, pp. 320–330.

— (2010). “Development informatics: Current status and future prospects.” In: *Exploring Success and Failure in Development Informatics: Innovation, Research and Practice. Proceedings of the 4th International IDIA Development Informatics Conference*. Ed. by Jacques Steyn. School of Information Technology, Monash South Africa.

Walsham, Geoff and Sundeep Sahay (2006). "Research on information systems in developing countries: current landscape and future prospects." In: *Information Technology for Development* 12 (1), pp. 7–24. ISSN: 0268-1102. DOI: 10.1002/itdj.20020. URL: http://portal.acm.org/citation.cfm?id=1146077.1146079.

Warsta, Juhani and Pekka Abrahamsson (2003). "Is Open Source Software Development Essentially an Agile Method?" In: *Proceedings of the 3rd Workshop on Open Source Software Engineering*. Ed. by Joseph Feller et al., pp. 143–147.

Weber, Ron (2004). "Editor's Comments: The Rhetoric of Positivism Versus Interpretivism: A Personal View." In: *MIS Quarterly* 28.1, pp. iii–xii.

— (2009). "Research on ICT for Development: Some Reflections on Rhetoric, Rigor, Reality, and Relevance." In: *Proceedings of the 3rd International IDIA Development Informatics Conference*. Ed. by Jacques Steyn. Keynote address. Berg-en-Dal, Kruger National Park, South Africa, pp. 2–27.

Weber, Steven (2004). *Open Source Software in Developing Economies*. URL: http://mediaresearchhub.ssrc.org/open-source-software-in-developing-economies/attachment (visited on Dec. 6, 2011).

Weerawarana, Sanjiva and Jivaka Weeratunge (2004). *Open source in Developing Countries*. Sida. URL: http://www.sida.se/publications (visited on Nov. 2, 2008).

West, Joel and Scott Gallagher (2006). "Challenges of open innovation: the paradox of firm investment in open-source software." In: *R&D Management* 36.3, pp. 319–331.

West, Joel and Siobhán O'Mahony (2005). "Contrasting Community Building in Sponsored and Community Founded Open Source Projects." In: *Proceedings of the 38th Annual Hawai'i International Conference on System Sciences*.

Wichmann, Thorsten (2002). *Free/Libre Open Source Software: Survey and Study, Final Report, Part 1: Use of Open Source Software in Firms and Public Institutions – Evidence from Germany, Sweden and UK*. Berlecon Research. URL: http://flossproject.org/report/ (visited on Dec. 2, 2011).

Widrow, Bernard, Reiner Hartenstein, and Robert Hecht-Nielsen (2005). *Eulogy: 1917 Karl Steinbuch 2005*. IEEE Computational Intelligence Society Newsletter.

Wilde, Thomas and Thomas Hess (2007). "Forschungsmethoden der Wirtschaftsinformatik - Eine empirische Untersuchung." In: *Wirtschaftsinformatik* 49.4, pp. 280–287.

Williams, Laurie and Alistair Cockburn (2003). "Agile Software Development: It's about Feedback and Change." In: *IEEE Computer* 36.6, pp. 39–43.

Winter, Robert (2008). "Design science research in Europe." In: *European Journal of Information Systems* 17, pp. 470–475.

World Bank (1999). *World Development Report: Knowledge for Development*. Oxford University Press.

Xia, Weidong and Gwanhoo Lee (2004). "Grasping the Complexity of IS Development Projects." In: *Communications of the ACM* 47.5, pp. 69–74.

Yang, Song, Heejin Lee, and Sherah Kurnia (2007). "Understanding the Relationship between Information and Communication Technology (ICT) and Social Capital: A Conceptual Framework." In: *Proceedings of the 13th Asia Pacific Management Conference*. Melbourne, Australia, pp. 231–241.

Yin, Robert K. (2009). *Case Study Research: Design and Methods (Applied Social Research Methods)*. 4th ed. Sage Publications, Inc. ISBN: 9781412960991. URL: `http://www.worldcat.org/isbn/1412960991`.

Young, Lincoln and Jonathan Hampshire (2000). *Promoting practical sustainability*. Canberra: Quality Assurance Group, Australian Agency for International Development. URL: `http://www.ausaid.gov.au/publications/pdf/sustainability.pdf` (visited on June 17, 2011).

Zurcher, F. W. and B. Randell (1968). "Iterative Multi-Level Modeling: A Methodology for Computer System Design." In: *Proceedings IFIP Congress*.

Ågerfalk, Pär J. et al. (2005). "A Framework for considering Opportunities and Threats in Distributed Software Development." In: *Proceedings of the International Workshop on Distributed Software Development (DiSD 2005)*. Austrian Computer Society. Paris.

Summary

This thesis is motivated by the poor track record of information system projects in the developing country context. Especially the sustainability of information systems is low. Many projects are donor driven due to the scarce local resources. But when donors end their commitment, information systems often get abandoned. Information system sustainability is an important, but neglected topic. In the Mozambican OPUS student information system project it became apparent that success and sustainability are hard to achieve, especially as there is little guidance from literature. The research in this thesis is an attempt to illuminate specific areas of relevance to information system projects in the context of scarce resources. The major outcome is a methodology to guide such projects, called the *appropriate information system development (AISD)* methodology.

The AISD methodology is based on research questions about several areas. First of all, there is the fundamental question how to make information system projects specifically appropriate for low resource contexts. The thesis takes the route towards so-called *appropriate technology* in order to pay respect to the target community or organization. Appropriate technology focuses on the use of locally available resources and on increased technological self-reliance. It has been applied to many areas over the last decades, like construction and architecture. For example, it would suggest to use hand pumps rather than a high tech irrigation system, because hand pumps are easier to maintain; in case of a breakdown of the high tech irrigation system it would have to be abandoned by a community that does not have access to the respective spare parts or the skills to carry out the sophisticated repair work. Appropriate technology is only recently being considered in the domain of ICT. Here, a simple example would be the preference for standard off-the-shelf computers instead of advanced rack-based servers. For the latter, repair may simply be impossible. Another example would be the 'one laptop per child' initiative, which tries to

provide simple, affordable computers for the masses. But appropriate technology also aims at increased local skills. This is also the approach of the AISD methodology: to combine the development of a product with learning the required skills to maintain the product in the long run.

Second, information system development requires specialized skills, which are often not available in a given context. Therefore, the AISD methodology makes the provision for less experienced actors to collaborate with more experienced partners. It supports the open source software development model and hence provides access to the source code of software systems. This provides a basic condition for learning by studying the source code. But access is not enough. In order to improve local technological capabilities, active, productive engagement with the technology development process is necessary. This is what the *effective use* principle is all about. In order to nurture local involvement, the local users and developers shall be involved early and frequently in evaluation and by taking responsibility for increasingly complex tasks.

Third, even if useful systems are developed, the user organization needs to prepare and adapt itself to accommodate and institutionalize it. This is a particular challenge for organizations that have little or no prior experience in information system projects. Information systems are often seen as primarily technical innovation. Therefore, if no accompanying effort is made to show users the utility for their work activities and to maintain management commitment, then the best system may be abandoned instantly. Managing this change process is another aspect that has been investigated.

Fourth, another piece in the puzzle of long-term success is the availability of support. Users as well as technicians need to have someone to get in contact with in case they have reached their wit's end. Support is cheaper the closer to the source of the problem it can be provided. Information about the conditions of a specific problem is for free at the place where the problem occurred. There is a price to be paid for transferring the information elsewhere. This price increases if issues like cultural distance between the source of the problem and the support service provider come into play. Therefore, a multilevel support structure is considered in this thesis. The support services to be offered by the service providers are based on community empowerment methods, which comprise activities in the categories of service delivery, capacity building, advocacy, and social mobilization. The support activities aim at improving the capabilities and the self-reliance of local organizations.

Fifth, the concepts concerning appropriate technology, open source software devel-

opment, organizational implementation, and empowerment based support have been applied practically in the OPUS project. For the evaluation of the effectiveness of their application, an analysis was made that covers the complete Mozambican OPUS project. The analysis was based on a variation of the sociological theory of structuration. The specific variation is a framework to analyze cross-cultural software production and use. The analysis was able to uncover inherent contradictions and different measures of success that the different participants held. It is not surprising that such differences exist in a project with partners from such diverse European and African backgrounds. The analysis was able to find a set of five success dimensions that are relevant to guide cross-cultural information system projects to less conflict and a higher probability of long-term success. The set of dimensions comprises: local contributions, shared resources, embedding, utility, and scalability.

Finally, the thesis integrates the insights from the five previous steps into the AISD methodology. The methodology is an example of a so called information system design theory. The goal of formulating the AISD methodology is to make the insights gained during the research more easily applicable to similar information system development projects. The methodology covers the entire system development life cycle. According to appropriate technology, it focuses on increased technological self-reliance and on effective use; it aims at two goals: (a) product development and (b) local learning. These goals are facilitated by a set of tools and methods. The description of the AISD methodology concludes with a set of theorems, which illustrate how the methodology responds to the research questions.

Samenvatting (Summary in Dutch)

De motivatie voor dit proefschrift is de lage succesratio van informatiesysteem-projecten in ontwikkelingslanden. Vooral de duurzaamheid van de informatiesystemen is problematisch. Veel projecten zijn donor-gestuurd vanwege de lokaal schaars beschikbare middelen. Maar zodra de betrokkenheid van de donoren wegvalt, dan komen ook de informatiesystemen vaak in het gedrang. De duurzaamheid van informatiesystemen is nochtans een belangrijk, maar relatief verwaarloosd onderwerp. In het Mozambicaanse "OPUS project", gericht op het implementeren van een student-informatiesysteem bij instellingen van hoger onderwijs, werd duidelijk dat het bereiken van succes en duurzaamheid een moeilijke opgave is, met name ook omdat er in de literatuur weinig richtlijnen terzake te vinden zijn. Het onderzoek in dit proefschrift is een poging specifieke gebieden te belichten die van belang zijn bij informatiesysteem-projecten in een context van schaarse middelen. De belangrijkste uitkomst is een methodologie om dergelijke projecten te begeleiden, de appropriate information system development (AISD) methodologie, gebaseerd op onderzoeksvragen vanuit verschillende gebieden en invalshoeken.

Ten eerste is er de fundamentele vraag hoe informatiesysteem-projecten het best ingericht kunnen worden in een context van schaarse hulpmiddelen. Het proefschrift volgt de insteek van de zogenaamde Appropriate technology waarin respect voor de (lokale) doelgroep of organisatie voorop staat. Meer bepaald richt appropriate technology zich op het gebruik van lokaal beschikbare middelen en op grotere technologische zelfredzaamheid. Het is afgelopen decennia toegepast op vele gebieden, zoals bouw en architectuur. Vanuit deze invalshoek zouden bijvoorbeeld in een watervoorzieningsproject handpompen aangeraden worden in plaats van een high-tech irrigatiesysteem, omdat handpompen relatief gemakkelijk te onderhouden zijn. Bij een defect van een high-tech irrigatiesysteem daarentegen is de lokale gemeenschap vrij hulpeloos daar deze geen toegang tot nieuwe onderdelen heeft of de

vaardigheden mist om verfijnde reparatiewerkzaamheden uit te voeren, met als gevolg dat het project dreigt teloor te gaan. Appropriate technology wordt pas sinds kort toegepast op het gebied van ICT. Een simpel voorbeeld is de aanschaf van standaard off-the-shelf computers in plaats van geavanceerde rack-based servers omdat voor deze laatste reparatie complexer is en daardoor in praktijk onmogelijk zou kunnen zijn. Een ander voorbeeld is het 'one laptop per child' project, een initiatief voor het beschikbaar stellen van eenvoudige, betaalbare computers aan grote groepen kinderen. Maar appropriate technology is ook gericht op verhoging van lokale vaardigheden. En dit is ook de benadering van de AISD methodologie: de juiste ontwikkeling van een product combineren met het leren van de vereiste vaardigheden om het product op lange termijn in stand te houden.

Ten tweede: informatiesysteem-ontwikkeling vraagt om specialistische vaardigheden, die vaak niet beschikbaar zijn in een bepaalde context. Daarom voorziet de AISD methodologie in de mogelijkheid voor minder ervaren actoren tot samenwerking met meer ervaren partners. AISD ondersteunt het open source software ontwikkelingsmodel van vrije toegang tot de broncode van softwaresystemen. Dit is een belangrijk aspect doordat partners op deze wijze praktijkgewijs kunnen leren door bestudering van de broncode. Maar toegang tot de code alleen is niet genoeg. Om de lokale technologische vaardigheden te verbeteren, is een actief en productief engagement met het technologie-ontwikkelingsproces noodzakelijk. Dit is de essentie van het effective use principe. Om de lokale betrokkenheid te bevorderen, is nodig dat lokale gebruikers en ontwikkelaars vanaf het begin regelmatig betrokken zijn bij de evaluatie van het ontwikkel- en implementatieproces en hun verantwoordelijkheid nemen m.b.t. de complexe taken die hiermee gemoeid zijn.

Ten derde, ontwikkeling van bruikbare systemen op zich is niet voldoende: ook de gebruikersorganisatie moet zich voorbereiden en aanpassen teneinde het systeem op een duurzame, geïnstitutionaliseerde manier te implementeren. Dit aspect is vooral een uitdaging voor organisaties met weinig of geen ervaring met informatiesysteem-projecten. Informatiesystemen worden door deze namelijk vaak gezien als een voornamelijk technische aangelegenheid. In deze gevallen mislukt implementatie van het systeem vaak omdat de bijbehorende maatregelen uitblijven om gebruikers te overtuigen van het nut van het systeem voor hun dagelijkse werkzaamheden, alsmede verzuimd wordt voldoende bestuurlijke ondersteuning voor het systeem te creëren. De organisatie en het management van dit bewustwordings- en veranderingsproces is een ander aspect dat is onderzocht.

Ten vierde, een stukje in de puzzel van lange-termijn succes is de beschikbaarheid

van voldoende ondersteuning. Gebruikers en technici hebben contactpersonen nodig tot wie ze zich kunnen wenden in het geval van problemen. Het verdient hierbij, vanuit kosten- en bereikbaarheidsoogpunt, de voorkeur om deze ondersteuning zo dichtbij mogelijk te organiseren. Informatie over de condities van een specifiek probleem zijn het best vast te stellen op de plaats waar het probleem zich heeft voorgedaan. De prijs van ondersteuning neemt alleen maar toe bij een grotere geografische en culturele afstand tussen de probleembron en de ondersteunende instantie. In dit verband wordt een meerlaagse ondersteuningsstructuur voorgesteld in dit proefschrift. Hierbij is het de bedoeling dat ondersteunende diensten worden aangeboden op basis van zogenaamde "community empowerment" methoden waarbij veel aandacht wordt besteed aan capaciteitsopbouw, belangenbehartiging en sociale mobilisatie. Op deze manier zijn de ondersteunende activiteiten gericht op verhoging van de zelfredzaamheid van de lokale organisaties.

Ten vijfde: het "OPUS project" in Mozambique heeft als casus gediend voor het toepassen van de hierboven vermelde concepten van appropriate technology, open source software ontwikkeling, organisatorische implementatie, en empowerment gebaseerde ondersteuning. Teneinde de effectiviteit van voornoemde aspecten na te gaan is een analyse uitgevoerd van het Mozambicaanse OPUS project, gebaseerd op een variant van de sociologische structuratietheorie die een kader biedt om productie en gebruik van cross-culturele software te analyseren. Met deze analyse konden inherente tegenstrijdigheden en verschillende maatstaven voor succes vanuit het oogpunt van de verschillende deelnemers blootgelegd worden. Het mag uiteraard geen verrassing heten dat zulke verschillen bestaan in een project met partners met zulke uiteenlopende Europese en Afrikaanse achtergronden. Met behulp van de analyse werden vijf succesdimensies gevonden voor de begeleiding van cross-culturele informatiesysteem-projecten gericht op reductie van conflicten en een hogere kans op succes op lange termijn. Deze dimensies bestaan uit: lokale bijdragen, gedeelde bronnen, inbedding, nut, en schaalbaarheid.

Ten slotte: de inzichten uit deze vijf voorgaande stappen van de AISD methodologie worden gecombineerd in dit proefschrift. De methodologie is een voorbeeld van een zogenaamde informatiesysteem- ontwerptheorie. Het doel van het formuleren van de AISD methodologie is om de inzichten opgedaan tijdens het onderzoek beschikbaar en bruikbaar te maken voor soortgelijke informatiesysteem-ontwikkelingsprojecten in de toekomst. De methodologie omvat de gehele ontwikkelingscyclus. Conform de appropriate technology filosofie richt de methodologie zich op grotere technologische zelfredzaamheid en effectief

gebruik waarbij (a) (participatie in) productontwikkeling en (b) lokaal leren als doelstellingen centraal staan. Deze doelstellingen worden ondersteund en vergemakkelijkt door een set van tools en methoden. De beschrijving van de AISD methodologie wordt afgesloten met een aantal stellingen, die illustreren hoe de methodologie antwoord geeft op de onderzoeksvragen.

Acknowledgements

Naturally, a major undertaking like writing a thesis cannot be done without the support of a range of people and organizations. The framework for this thesis would not have been set up in the way it was without the initiative of my co-supervisor Ed Simons. This was a beneficial setup that made it possible to investigate a topic of immediate practical relevance. My supervisor Theo van der Weide has guided me successfully during the different stages of writing, despite the geographical distance, with his innovative and approachable style. It is good to know professors like Theo who put such a high value on his students. Over the years, Monique in het Veld was a great companion throughout the OPUS project and has shown that creative, open minds from diverse backgrounds can have a lot to contribute to information system projects, especially when meeting different cultures.

It was a pleasure to meet so many warm hearted colleagues in Africa, first and foremost the people at the Catholic University of Mozambique (UCM). Former vice-rector Padre Francisco Ponsi made me feel welcome right from the beginning – and sorted out all practical issues when necessary; I won't forget the travel to a conference, where I was only able to board the plane from Mozambique to South Africa because Padre Ponsi made every effort to obtain the travel documents from the immigration offices, in which he succeeded two hours before departure. Rector Padre Alberto Ferreira acknowledged the relevance of the research publications for the UCM. Despite all the responsibility weighing on his shoulders, he has remained approachable. Sister Raffaella occupies a place of special relevance in the OPUS implementation at UCM. She kept her faith and made the most out of our common effort. My 'pupils' from the IS department, Stelio Macumbe and João Pereira, made an effort to grow and support OPUS over the years.

Seven years in Mozambique means seven years of support by the Austrian NGO HORIZONT3000. It was in the preparatory course organized by HORIZONT3000 in Vienna

that the issue of sustainability of technical assistance was made a point of discussion. This had a strong impact on the topic of investigation of the thesis. Andrea Heiden played an important role in my continued involvement. HORIZONT3000 also facilitated a great deal in matters concerning life abroad with small children.

Last but not least, Nuffic, the Netherlands organization for international cooperation in higher education, was an important enabler of OPUS related activities. Without the support of Nuffic, the OPUS project would never have started off for the benefit of several African universities. Not only was the initial development being nurtured in Mozambique, but support continued so that an open source license could be applied, which makes OPUS accessible to many more universities around the globe.

Curriculum Vitae

Markus Pscheidt was born in Graz, Austria on July 12, 1975. He studied Telematik at Graz University of Technology from 1994 to 2000. Part of the study included a one-year stay from 1997 to 1998 at Luleå University of Technology, Sweden. The master thesis, called "Secure multimedia conferences", applied cryptography on video conferencing systems.

His work experience includes the production and use of information systems in various fields. From 2000 to 2005 Markus was responsible for leading the technical development of the medical laboratory information system "datalabX" for Bartelt GmbH, Graz. From 2003 to 2004 he was involved in the development of the "European Social Database" at Weberhofer GmbH, Vienna. Furthermore, Markus periodically acted as an instructor for various aspects of object oriented software development.

In 2005, Markus started an assignment as an international development cooperation consultant for the Austrian NGO HORIZONT3000 at the Catholic University in Mozambique. The main objective was to develop the OPUS student information system for Mozambican and later also Zambian universities. Since 2008 Markus is an external PhD student at the Radboud University Nijmegen in the Netherlands. The thesis is related to overcoming typical challenges encountered in the collaboration between globally distributed development teams in a sustainable way.